

(43) Date of publication:
31.03.2021 Bulletin 2021/13

(51) Int Cl.: **G05B 19/042** ^(2006.01) **G05B 19/05** ^(2006.01)
G06F 8/34 ^(2018.01)

(21) Application number: **20166932.2**

(22) Date of filing: **31.03.2020**

(84) Designated Contracting States:
**AL AT BE BG CH CY CZ DE DK EE ES FI FR GB
 GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO
 PL PT RO RS SE SI SK SM TR**
 Designated Extension States:
BA ME
 Designated Validation States:
KH MA MD TN

- **STUMP, Andrew R.**
Mayfield Heights, OH 44124 (US)
- **CARRARA, Anthony**
Mayfield Heights, OH 44124 (US)
- **SRINIVASAN, Eashwer**
Mayfield Heights, OH 44124 (US)
- **COMO, Christopher W.**
Mayfield Heights, OH 44124 (US)
- **BILLI-DURAN, Sharon M.**
Mayfield Heights, OH 44124 (US)

(30) Priority: 27.09.2019 US 201916585985

(71) Applicant: **Rockwell Automation Technologies, Inc.**
Mayfield Heights, OH 44124-6188 (US)

(74) Representative: **Grünecker Patent- und
Rechtsanwälte
PartG mbB
Leopoldstraße 4
80802 München (DE)**

(72) Inventors:

- **ERICSSON, Matthew R.**
Mayfield Heights, OH 44124 (US)

(54) **PREFERENTIAL AUTOMATION VIEW CURATION**

(57) An industrial integrated development environment (IDE) comprises a development interface that affords a user a great deal of control over the editing tools, workspace canvases, and project information rendered at a given time. The industrial IDE system automatically filters the tools, panels, and information available for selection based on a current project development task, such that a focused subset of editing tools relevant to a current development task or context are made available

for selection while other tools are hidden. The development interface also allows the user to selectively render or hide selected tools or information from among the relevant, filtered set of tools. This can reduce or eliminate unnecessary clutter and aid in quickly and easily locating and selecting a desired editing function. The IDE's development interface can also conform to a structured organization of workspace canvases and panels that facilitates intuitive workflow.



Description

BACKGROUND

[0001] The subject matter disclosed herein relates generally to industrial automation systems, and, for example, to industrial programming development platforms.

BRIEF DESCRIPTION

[0002] The following presents a simplified summary in order to provide a basic understanding of some aspects described herein. This summary is not an extensive overview nor is intended to identify key/critical elements or to delineate the scope of the various aspects described herein. Its sole purpose is to present some concepts in a simplified form as a prelude to the more detailed description that is presented later.

[0003] In one or more embodiments, a system for developing industrial applications is provided, comprising a user interface component configured to render an industrial integrated development environment (IDE) development interface and to receive, via interaction with the development interface, industrial design input that defines aspects of an industrial automation project, wherein the development interface comprises one or more workspace canvases configured to facilitate development of a selected aspect of the industrial automation project, and a global panel control bar comprising visibility icons corresponding to respective panels available to be invoked on the development interface; and a project generation component configured to generate system project data based on the industrial design input, wherein the interface display comprises a left global panel area, a right global panel area, and a bottom global panel area, the respective panels are designated to one of the left global panel area, the right global panel area, or the bottom global panel area, and the user interface component is further configured to, in response to selection of a visibility icon, of the visibility icons, toggle a visibility of a panel corresponding to the visibility icon in one of the left global panel area, the right global panel area, or the bottom global panel area.

[0004] Also, one or more embodiments provide a method for curating an industrial development workspace, comprising displaying, by an industrial integrated development environment (IDE) system comprising a processor, a development interface on a client device, wherein the displaying comprises: displaying one or more workspace canvases on which respective development tasks are performed, and displaying a global panel control bar comprising visibility icons corresponding to respective panels available to be invoked on the development interface; receiving, by the industrial IDE system, selection of a visibility icon, of the visibility icons, via interaction with the development interface; and in response to the receiving, toggling, by the industrial IDE system, a visi-

bility of a panel corresponding to the visibility icon, wherein the toggling comprises adding the panel to or removing the panel from one of a left global panel area, a right global panel area, or a bottom global panel area of the development interface.

[0005] Also, according to one or more embodiments, a non-transitory computer-readable medium is provided having stored thereon instructions that, in response to execution, cause an industrial integrated development environment (IDE) system to perform operations, the operations comprising displaying a development interface for the industrial IDE system on a client device, wherein the displaying comprises: displaying one or more workspace canvases on which respective development tasks are performed, and displaying a global panel control bar comprising visibility icons corresponding to respective panels available to be invoked on the development interface; receiving selection of a visibility icon, of the visibility icons, via interaction with the development interface; and in response to the receiving, toggling a visibility of a panel corresponding to the visibility icon, wherein the toggling comprises adding the panel to or removing the panel from one of a left global panel area, a right global panel area, or a bottom global panel area of the development interface.

[0006] To the accomplishment of the foregoing and related ends, certain illustrative aspects are described herein in connection with the following description and the annexed drawings. These aspects are indicative of various ways which can be practiced, all of which are intended to be covered herein. Other advantages and novel features may become apparent from the following detailed description when considered in conjunction with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007]

FIG. 1 is a block diagram of an example industrial control environment.

FIG. 2 is a block diagram of an example integrated development environment (IDE) system.

FIG. 3 is a diagram illustrating a generalized architecture of an industrial IDE system.

FIG. 4 is a diagram illustrating several example automation object properties that can be leveraged by the IDE system in connection with building, deploying, and executing a system project.

FIG. 5 is a diagram illustrating example data flows associated with creation of a system project for an automation system being designed using an industrial IDE system.

FIG. 6 is a diagram illustrating an example system project that incorporates automation objects into a project model.

FIG. 7 is a diagram illustrating commissioning of a system project.

FIG. 8 is a diagram illustrating an example architecture in which cloud-based IDE services are used to develop and deploy industrial applications to a plant environment.

FIG. 9 is an example development interface that can be rendered by one or more embodiments of an industrial IDE system's user interface component.

FIG. 10a is a close-up view of a global panel control bar illustrating an example organization of panel visibility icons.

FIG. 10b is an example View menu that can be rendered as a drop-down menu in response to selection of a View option in a menu bar of an industrial IDE system.

FIG. 11a is a view of a top right corner of a development interface depicting a Properties panel pinned in a right global panel area.

FIG. 11b is a view of the top right corner of the development interface depicting selection of an Online panel as an overlaid panel in the right global panel area.

FIG. 11c is a view of the top right corner of the development interface depicting two pinned panels that are visible simultaneously.

FIG. 11d is a view of the top right corner of the development interface in which a Toolbox panel is rendered as an overlay above a Properties panel.

FIG. 11e is a view of the top right corner of the development interface in which a Toolbox panel is switched to be a pinned panel.

FIG. 12 is a view of the top right corner of the development interface depicting a panel drop area for a right global panel area.

FIG. 13a is a view of two horizontally stacked pinned panels in a default non-collapsed state.

FIG. 13b is a view of the two horizontally stacked pinned panels in which the lower panel is in a collapsed state.

FIG. 13c is a view of the two horizontally stacked pinned panels in which the upper panel is in a collapsed state.

FIG. 14 is a view of an example canvas within a canvas area of an industrial IDE development interface.

FIG. 15 is a view of an industrial development interface in which two canvases have been stacked horizontally.

FIG. 16a is a view of two tabbed development interfaces in which one tab is selected, causing the corresponding ladder logic canvas to be rendered in the canvas area.

FIG. 16b is a view of two tabbed development interfaces in which one tab is selected, causing the corresponding tag database canvas to be rendered in the canvas area.

FIG. 17a is a view of a development interface in which a single canvas is open and no left, right, or bottom panels are invoked.

FIG. 17b is a view of the development interface in

which an Explorer panel has been rendered visible in a left global panel area and a Properties panel has been rendered in a right global panel area.

FIG. 17c is a view of the development interface in which a Layers panel has been added to the previous view.

FIG. 17d a view of the development interface in which adds a second canvas stacked horizontally with a pre-existing canvas.

FIG. 18 is a view of an Explorer panel, which resides in a left global panel area of a development interface when invoked.

FIG. 19a is a view of the Explorer panel with the Logical System view currently selected.

FIG. 19b is a view of the Explorer panel with the Execution System view currently selected.

FIG. 20 is an example Explorer panel depicting a System navigation tree for an example automation system project

FIG. 21a illustrates an example response of an industrial IDE development interface when a user selects, but does not launch, a ladder logic node representing a ladder logic program of the system project.

FIG. 21b illustrates an example response of the industrial IDE development interface when a user launches the ladder logic node 2002.

FIG. 21c illustrates an example response of the industrial IDE development interface when a user right-clicks on the ladder logic node.

FIG. 22a is a view of the Explorer panel with the Application view and the Controller tab currently selected.

FIG. 22b is a view of the Explorer panel with the Application view and the HMI tab currently selected.

FIG. 23 is a view of an industrial IDE workspace canvas on which a portion of an example structure text program is rendered in response to selection of a structured text application node.

FIG. 24 is a view of an industrial IDE workspace canvas on which a portion of an example function block diagram program is rendered in response to selection of a function block diagram application node.

FIG. 25 is a view of an Explorer panel with the Devices view currently selected.

FIG. 26 is a view of an industrial IDE workspace canvas on which information for an example controller is rendered in response to selection of a controller node.

FIG. 27 is a view of an Explorer panel with the Library view currently selected.

FIG. 28 is a view of an Explorer panel with the Extensions view currently selected.

FIG. 29a is a left-side instance of an industrial IDE development interface that is distributed across two display devices.

FIG. 29b is a right-side instance of the industrial IDE development interface that is distributed across two

display devices.

FIG. 30 is an example Available Tabs menu.

FIG. 31a is an industrial IDE development interface rendered in accordance with a first layout mode suitable for scenarios in which there are no width restrictions.

FIG. 31b is an industrial IDE development interface rendered in accordance with a second layout mode that is invoked when the available screen width is below a first threshold width.

FIG. 31c is an industrial IDE development interface rendered in accordance with a third layout mode that may be initiated when the available screen width is below a second threshold width that is smaller than the first threshold width.

FIG. 32a is a flowchart of a first part of an example methodology for customizing panel visibility and layout on a development interface of an industrial IDE system.

FIG. 32b is a flowchart of a second part of the example methodology for customizing panel visibility and layout on the development interface of the industrial IDE system.

FIG. 32c is a flowchart of a third part of the example methodology for customizing panel visibility and layout on the development interface of the industrial IDE system.

FIG. 33a is a flowchart of a first part of an example methodology for browsing and rendering aspects of an industrial automation project via interaction with an industrial IDE development interface.

FIG. 33b is a flowchart of a second part of the example methodology for browsing and rendering aspects of the industrial automation project via interaction with the industrial IDE development interface.

FIG. 34a is a flowchart of a first part of an example methodology for manipulating workspace canvases within an industrial IDE development interface.

FIG. 34b is a flowchart of a second part of the example methodology for manipulating workspace canvases within the industrial IDE development interface.

FIG. 34c is a flowchart of a third part of the example methodology for manipulating workspace canvases within the industrial IDE development interface.

FIG. 35a is a flowchart of a first part of an example methodology for automatically curating a set of available project editing tools by an industrial IDE development interface based on a current development task being performed by a user.

FIG. 35b is a flowchart of a second part of the example methodology for automatically curating the set of available project editing tools by the industrial IDE development interface based on the current development task being performed by the user.

FIG. 36 is an example computing environment.

FIG. 37 is an example networking environment.

DETAILED DESCRIPTION

[0008] The subject disclosure is now described with reference to the drawings, wherein like reference numerals are used to refer to like elements throughout. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding thereof. It may be evident, however, that the subject disclosure can be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to facilitate a description thereof.

[0009] As used in this application, the terms "component," "system," "platform," "layer," "controller," "terminal," "station," "node," "interface" are intended to refer to a computer-related entity or an entity related to, or that is part of, an operational apparatus with one or more specific functionalities, wherein such entities can be either hardware, a combination of hardware and software, software, or software in execution. For example, a component can be, but is not limited to being, a process running on a processor, a processor, a hard disk drive, multiple storage drives (of optical or magnetic storage medium) including affixed (e.g., screwed or bolted) or removable affixed solid-state storage drives; an object; an executable; a thread of execution; a computer-executable program, and/or a computer. By way of illustration, both an application running on a server and the server can be a component. One or more components can reside within a process and/or thread of execution, and a component can be localized on one computer and/or distributed between two or more computers. Also, components as described herein can execute from various computer readable storage media having various data structures stored thereon. The components may communicate via local and/or remote processes such as in accordance with a signal having one or more data packets (e.g., data from one component interacting with another component in a local system, distributed system, and/or across a network such as the Internet with other systems via the signal). As another example, a component can be an apparatus with specific functionality provided by mechanical parts operated by electric or electronic circuitry which is operated by a software or a firmware application executed by a processor, wherein the processor can be internal or external to the apparatus and executes at least a part of the software or firmware application. As yet another example, a component can be an apparatus that provides specific functionality through electronic components without mechanical parts, the electronic components can include a processor therein to execute software or firmware that provides at least in part the functionality of the electronic components. As further yet another example, interface(s) can include input/output (I/O) components as well as associated processor, application, or Application Programming Interface (API) components. While the foregoing examples are directed to aspects of a component, the exemplified aspects or features also

apply to a system, platform, interface, layer, controller, terminal, and the like.

[0010] As used herein, the terms "to infer" and "inference" refer generally to the process of reasoning about or inferring states of the system, environment, and/or user from a set of observations as captured via events and/or data. Inference can be employed to identify a specific context or action, or can generate a probability distribution over states, for example. The inference can be probabilistic—that is, the computation of a probability distribution over states of interest based on a consideration of data and events. Inference can also refer to techniques employed for composing higher-level events from a set of events and/or data. Such inference results in the construction of new events or actions from a set of observed events and/or stored event data, whether or not the events are correlated in close temporal proximity, and whether the events and data come from one or several event and data sources.

[0011] In addition, the term "or" is intended to mean an inclusive "or" rather than an exclusive "or." That is, unless specified otherwise, or clear from the context, the phrase "X employs A or B" is intended to mean any of the natural inclusive permutations. That is, the phrase "X employs A or B" is satisfied by any of the following instances: X employs A; X employs B; or X employs both A and B. In addition, the articles "a" and "an" as used in this application and the appended claims should generally be construed to mean "one or more" unless specified otherwise or clear from the context to be directed to a singular form.

[0012] Furthermore, the term "set" as employed herein excludes the empty set; e.g., the set with no elements therein. Thus, a "set" in the subject disclosure includes one or more elements or entities. As an illustration, a set of controllers includes one or more controllers; a set of data resources includes one or more data resources; etc. Likewise, the term "group" as utilized herein refers to a collection of one or more entities; e.g., a group of nodes refers to one or more nodes.

[0013] Various aspects or features will be presented in terms of systems that may include a number of devices, components, modules, and the like. It is to be understood and appreciated that the various systems may include additional devices, components, modules, etc. and/or may not include all of the devices, components, modules etc. discussed in connection with the figures. A combination of these approaches also can be used.

[0014] FIG. 1 is a block diagram of an example industrial control environment 100. In this example, a number of industrial controllers 118 are deployed throughout an industrial plant environment to monitor and control respective industrial systems or processes relating to product manufacture, machining, motion control, batch processing, material handling, or other such industrial functions. Industrial controllers 118 typically execute respective control programs to facilitate monitoring and control of industrial devices 120 making up the controlled industrial assets or systems (e.g., industrial machines).

One or more industrial controllers 118 may also comprise a soft controller executed on a personal computer or other hardware platform, or on a cloud platform. Some hybrid devices may also combine controller functionality with other functions (e.g., visualization). The control programs executed by industrial controllers 118 can comprise substantially any type of code capable of processing input signals read from the industrial devices 120 and controlling output signals generated by the industrial controllers 118, including but not limited to ladder logic, sequential function charts, function block diagrams, or structured text.

[0015] Industrial devices 120 may include both input devices that provide data relating to the controlled industrial systems to the industrial controllers 118, and output devices that respond to control signals generated by the industrial controllers 118 to control aspects of the industrial systems. Example input devices can include telemetry devices (e.g., temperature sensors, flow meters, level sensors, pressure sensors, etc.), manual operator control devices (e.g., push buttons, selector switches, etc.), safety monitoring devices (e.g., safety mats, safety pull cords, light curtains, etc.), and other such devices. Output devices may include motor drives, pneumatic actuators, signaling devices, robot control inputs, valves, pumps, and the like.

[0016] Industrial controllers 118 may communicatively interface with industrial devices 120 over hardwired or networked connections. For example, industrial controllers 118 can be equipped with native hardwired inputs and outputs that communicate with the industrial devices 120 to effect control of the devices. The native controller I/O can include digital I/O that transmits and receives discrete voltage signals to and from the field devices, or analog I/O that transmits and receives analog voltage or current signals to and from the devices. The controller I/O can communicate with a controller's processor over a backplane such that the digital and analog signals can be read into and controlled by the control programs. Industrial controllers 118 can also communicate with industrial devices 120 over a network using, for example, a communication module or an integrated networking port. Exemplary networks can include the Internet, intranets, Ethernet, DeviceNet, ControlNet, Data Highway and Data Highway Plus (DH/DH+), Remote I/O, Fieldbus, Modbus, Profibus, wireless networks, serial protocols, and the like. The industrial controllers 118 can also store persisted data values that can be referenced by their associated control programs and used for control decisions, including but not limited to measured or calculated values representing operational states of a controlled machine or process (e.g., tank levels, positions, alarms, etc.) or captured time series data that is collected during operation of the automation system (e.g., status information for multiple points in time, diagnostic occurrences, etc.). Similarly, some intelligent devices - including but not limited to motor drives, instruments, or condition monitoring modules - may store data values that are used for control

and/or to visualize states of operation. Such devices may also capture time-series data or events on a log for later retrieval and viewing.

[0017] Industrial automation systems often include one or more human-machine interfaces (HMIs) 114 that allow plant personnel to view telemetry and status data associated with the automation systems, and to control some aspects of system operation. HMIs 114 may communicate with one or more of the industrial controllers 118 over a plant network 116, and exchange data with the industrial controllers to facilitate visualization of information relating to the controlled industrial processes on one or more pre-developed operator interface screens. HMIs 114 can also be configured to allow operators to submit data to specified data tags or memory addresses of the industrial controllers 118, thereby providing a means for operators to issue commands to the controlled systems (e.g., cycle start commands, device actuation commands, etc.), to modify setpoint values, etc. HMIs 114 can generate one or more display screens through which the operator interacts with the industrial controllers 118, and thereby with the controlled processes and/or systems. Example display screens can visualize present states of industrial systems or their associated devices using graphical representations of the processes that display metered or calculated values, employ color or position animations based on state, render alarm notifications, or employ other such techniques for presenting relevant data to the operator. Data presented in this manner is read from industrial controllers 118 by HMIs 114 and presented on one or more of the display screens according to display formats chosen by the HMI developer. HMIs may comprise fixed location or mobile devices with either user-installed or pre-installed operating systems, and either user-installed or pre-installed graphical application software.

[0018] Some industrial environments may also include other systems or devices relating to specific aspects of the controlled industrial systems. These may include, for example, a data historian 110 that aggregates and stores production information collected from the industrial controllers 118 or other data sources, device documentation stores containing electronic documentation for the various industrial devices making up the controlled industrial systems, inventory tracking systems, work order management systems, repositories for machine or process drawings and documentation, vendor product documentation storage, vendor knowledgebases, internal knowledgebases, work scheduling applications, or other such systems, some or all of which may reside on an office network 108 of the industrial environment.

[0019] Higher-level systems 126 may carry out functions that are less directly related to control of the industrial automation systems on the plant floor, and instead are directed to long term planning, high-level supervisory control, analytics, reporting, or other such high-level functions. These systems 126 may reside on the office network 108 at an external location relative to the plant fa-

cility, or on a cloud platform with access to the office and/or plant networks. Higher-level systems 126 may include, but are not limited to, cloud storage and analysis systems, big data analysis systems, manufacturing execution systems, data lakes, reporting systems, etc. In some scenarios, applications running at these higher levels of the enterprise may be configured to analyze control system operational data, and the results of this analysis may be fed back to an operator at the control system or directly to a controller 118 or device 120 in the control system.

[0020] The various control, monitoring, and analytical devices that make up an industrial environment must be programmed or configured using respective configuration applications specific to each device. For example, industrial controllers 118 are typically configured and programmed using a control programming development application such as a ladder logic editor (e.g., executing on a client device 124). Using such development platforms, a designer can write control programming (e.g., ladder logic, structured text, function block diagrams, etc.) for carrying out a desired industrial sequence or process and download the resulting program files to the controller 118. Separately, developers design visualization screens and associated navigation structures for HMIs 114 using an HMI development platform (e.g., executing on client device 122) and download the resulting visualization files to the HMI 114. Some industrial devices 120 - such as motor drives, telemetry devices, safety input devices, etc. - may also require configuration using separate device configuration tools (e.g., executing on client device 128) that are specific to the device being configured. Such device configuration tools may be used to set device parameters or operating modes (e.g., high/low limits, output signal formats, scale factors, energy consumption modes, etc.).

[0021] The necessity of using separate configuration tools to program and configure disparate aspects of an industrial automation system results in a piecemeal design approach whereby different but related or overlapping aspects of an automation system are designed, configured, and programmed separately on different development environments. For example, a motion control system may require an industrial controller to be programmed and a control loop to be tuned using a control logic programming platform, a motor drive to be configured using another configuration platform, and an associated HMI to be programmed using a visualization development platform. Related peripheral systems - such as vision systems, safety systems, etc. - may also require configuration using separate programming or development applications.

[0022] This segregated development approach can also necessitate considerable testing and debugging efforts to ensure proper integration of the separately configured system aspects. In this regard, intended data interfacing or coordinated actions between the different system aspects may require significant debugging due

to a failure to properly coordinate disparate programming efforts.

[0023] Industrial development platforms are also limited in terms of the development interfaces offered to the user to facilitate programming and configuration. These interfaces typically offer a fixed user experience that requires the user to develop control code, visualizations, or other control system aspects using a relatively fixed set of development interfaces. In many development scenarios, the number of editing options - e.g., function buttons or other selectable editing controls, configuration fields, etc. - that are displayed on the development platform's interface exceed the number required by the developer for a current project development task, resulting in an unnecessarily cluttered development workspace and rendering it difficult to locate a desired editing option.

[0024] To address at least some of these or other issues, one or more embodiments described herein provide an integrated development environment (IDE) for designing, programming, and configuring multiple aspects of an industrial automation system using a common design environment and data model. Embodiments of the industrial IDE can be used to configure and manage automation system devices in a common way, facilitating integrated, multi-discipline programming of control, visualization, and other aspects of the control system.

[0025] In some embodiments, the development interface rendered by the IDE system can afford the user a great deal of control over the editing tools, workspace canvases, and project information rendered at a given time. The IDE system also automatically filters the tools, panels, and information available for selection based on a determination of the current project development task being carried out by the user, such that a focused subset of editing tools relevant to a current development task are made available for selection while other tools are hidden. The development interface also allows the user to selectively render or hide selected tools or information from among the relevant, filtered set of tools. This approach can reduce or eliminate unnecessary clutter and assist the developer in quickly and easily locating and selecting a desired editing function. The IDE's development interface can also conform to a structured organization of workspace canvases and panels that facilitates intuitive workflow.

[0026] FIG. 2 is a block diagram of an example integrated development environment (IDE) system 202 according to one or more embodiments of this disclosure. Aspects of the systems, apparatuses, or processes explained in this disclosure can constitute machine-executable components embodied within machine(s), e.g., embodied in one or more computer-readable mediums (or media) associated with one or more machines. Such components, when executed by one or more machines, e.g., computer(s), computing device(s), automation device(s), virtual machine(s), etc., can cause the machine(s) to perform the operations described.

[0027] IDE system 202 can include a user interface

component 204 including an IDE editor 224, a project generation component 206, a project deployment component 208, one or more processors 218, and memory 220. In various embodiments, one or more of the user interface component 204, project generation component 206, project deployment component 208, the one or more processors 218, and memory 220 can be electrically and/or communicatively coupled to one another to perform one or more of the functions of the IDE system 202. In some embodiments, components 204, 206, and 208 can comprise software instructions stored on memory 220 and executed by processor(s) 218. IDE system 202 may also interact with other hardware and/or software components not depicted in FIG. 2. For example, processor(s) 218 may interact with one or more external user interface devices, such as a keyboard, a mouse, a display monitor, a touchscreen, or other such interface devices.

[0028] User interface component 204 can be configured to receive user input and to render output to the user in any suitable format (e.g., visual, audio, tactile, etc.). In some embodiments, user interface component 204 can be configured to communicatively interface with an IDE client that executes on a client device (e.g., a laptop computer, tablet computer, smart phone, etc.) that is communicatively connected to the IDE system 202 (e.g., via a hardwired or wireless connection). The user interface component 204 can then receive user input data and render output data via the IDE client. In other embodiments, user interface component 314 can be configured to generate and serve development interface screens to a client device (e.g., program development screens), and exchange data via these interface screens. As will be described in more detail herein, the development interfaces rendered by the user interface component 204 support a number of user experience features that simplify project development workflow, reduce stress associated with an overcluttered development workspace, and assist developers to locate desired editing functions more quickly and easily. Input data that can be received via various embodiments of user interface component 204 can include, but is not limited to, programming code, industrial design specifications or goals, engineering drawings, AR/VR input, DSL definitions, video or image data, or other such input. Output data rendered by various embodiments of user interface component 204 can include program code, programming feedback (e.g., error and highlighting, coding suggestions, etc.), programming and visualization development screens, etc.

[0029] Project generation component 206 can be configured to create a system project comprising one or more project files based on design input received via the user interface component 204, as well as industrial knowledge, predefined code modules and visualizations, and automation objects 222 maintained by the IDE system 202. Project deployment component 208 can be configured to commission the system project created by the project generation component 206 to appropriate indus-

trial devices (e.g., controllers, HMI terminals, motor drives, AR/VR systems, etc.) for execution. To this end, project deployment component 208 can identify the appropriate target devices to which respective portions of the system project should be sent for execution, translate these respective portions to formats understandable by the target devices, and deploy the translated project components to their corresponding devices.

[0030] The one or more processors 218 can perform one or more of the functions described herein with reference to the systems and/or methods disclosed. Memory 220 can be a computer-readable storage medium storing computer-executable instructions and/or information for performing the functions described herein with reference to the systems and/or methods disclosed.

[0031] FIG. 3 is a diagram illustrating a generalized architecture of the industrial IDE system 202 according to one or more embodiments. Industrial IDE system 202 can implement a common set of services and workflows spanning not only design, but also commissioning, operation, and maintenance. In terms of design, the IDE system 202 can support not only industrial controller programming and HMI development, but also sizing and selection of system components, device/system configuration, AR/VR visualizations, and other features. The IDE system 202 can also include tools that simplify and automate commissioning of the resulting project and assist with subsequent administration of the deployed system during runtime.

[0032] Embodiments of the IDE system 202 that are implemented on a cloud platform also facilitate collaborative project development whereby multiple developers 304 contribute design and programming input to a common automation system project 302. Collaborative tools supported by the IDE system can manage design contributions from the multiple contributors and perform version control of the aggregate system project 302 to ensure project consistency.

[0033] Based on design and programming input from one or more developers 304, IDE system 202 generates a system project 302 comprising one or more project files. The system project 302 encodes one or more of control programming; HMI, AR, and/or VR visualizations; device or sub-system configuration data (e.g., drive parameters, vision system configurations, telemetry device parameters, safety zone definitions, etc.); or other such aspects of an industrial automation system being designed. IDE system 202 can identify the appropriate target devices 306 on which respective aspects of the system project 302 should be executed (e.g., industrial controllers, HMI terminals, variable frequency drives, safety devices, etc.), translate the system project 302 to executable files that can be executed on the respective target devices, and deploy the executable files to their corresponding target devices 306 for execution, thereby commissioning the system project 302 to the plant floor for implementation of the automation project.

[0034] To support enhanced development capabilities,

some embodiments of IDE system 202 can be built on an object-based data model rather than a tag-based architecture. Automation objects 222 serve as the building block for this object-based development architecture. FIG. 4 is a diagram illustrating several example automation object properties that can be leveraged by the IDE system 202 in connection with building, deploying, and executing a system project 302. Automation objects 222 can be created and augmented during design, integrated into larger data models, and consumed during runtime. These automation objects 222 provide a common data structure across the IDE system 202 and can be stored in an object library (e.g., part of memory 220) for reuse. The object library can store predefined automation objects 222 representing various classifications of real-world industrial assets 402, including but not limited to pumps, tanks, valves, motors, motor drives (e.g., variable frequency drives), industrial robots, actuators (e.g., pneumatic or hydraulic actuators), or other such assets. Automation objects 222 can represent elements at substantially any level of an industrial enterprise, including individual devices, machines made up of many industrial devices and components (some of which may be associated with their own automation objects 222), and entire production lines or process control systems.

[0035] An automation object 222 for a given type of industrial asset can encode such aspects as 2D or 3D visualizations, alarms, control coding (e.g., logic or other type of control programming), analytics, startup procedures, testing protocols, validation reports, simulations, schematics, security protocols, and other such properties associated with the industrial asset 402 represented by the object 222. Automation objects 222 can also be geotagged with location information identifying the location of the associated asset. During runtime of the system project 302, the automation object 222 corresponding to a given real-world asset 402 can also record status or operational history data for the asset. In general, automation objects 222 serve as programmatic representations of their corresponding industrial assets 402, and can be incorporated into a system project 302 as elements of control code, a 2D or 3D visualization, a knowledgebase or maintenance guidance system for the industrial assets, or other such aspects.

[0036] FIG. 5 is a diagram illustrating example data flows associated with creation of a system project 302 for an automation system being designed using IDE system 202 according to one or more embodiments. A client device 504 (e.g., a laptop computer, tablet computer, desktop computer, mobile device, wearable AR/VR appliance, etc.) executing an IDE client application 514 can access the IDE system's project development tools and leverage these tools to create a comprehensive system project 302 for an automation system being developed. Through interaction with the system's user interface component 204, developers can submit design input 512 to the IDE system 202 in various supported formats, including industry-specific control programming (e.g., control

logic, structured text, sequential function charts, etc.) and HMI screen configuration input. Based on this design input 512 and information stored in an industry knowledge-base (predefined code modules 508 and visualizations 510, guardrail templates 506, physics-based rules 516, etc.), user interface component 204 renders design feedback 518 designed to assist the developer in connection with developing a system project 302 for configuration, control, and visualization of an industrial automation system.

[0037] In addition to control programming and visualization definitions, some embodiments of IDE system 202 can be configured to receive digital engineering drawings (e.g., computer-aided design (CAD) files) as design input 512. In such embodiments, project generation component 206 can generate portions of the system project 302 - e.g., by automatically generating control and/or visualization code - based on analysis of existing design drawings. Drawings that can be submitted as design input 512 can include, but are not limited to, P&ID drawings, mechanical drawings, flow diagrams, or other such documents. For example, a P&ID drawing can be imported into the IDE system 202, and project generation component 206 can identify elements (e.g., tanks, pumps, etc.) and relationships therebetween conveyed by the drawings. Project generation component 206 can associate or map elements identified in the drawings with appropriate automation objects 222 corresponding to these elements (e.g., tanks, pumps, etc.) and add these automation objects 222 to the system project 302. The device-specific and asset-specific automation objects 222 include suitable code and visualizations to be associated with the elements identified in the drawings. In general, the IDE system 202 can examine one or more different types of drawings (mechanical, electrical, piping, etc.) to determine relationships between devices, machines, and/or assets (including identifying common elements across different drawings) and intelligently associate these elements with appropriate automation objects 222, code modules 508, and/or visualizations 510. The IDE system 202 can leverage physics-based rules 516 as well as pre-defined code modules 508 and visualizations 510 as necessary in connection with generating code or project data for system project 302.

[0038] The IDE system 202 can also determine whether pre-defined visualization content is available for any of the objects discovered in the drawings and generate appropriate HMI screens or AR/VR content for the discovered objects based on these pre-defined visualizations. To this end, the IDE system 202 can store industry-specific, asset-specific, and/or application-specific visualizations 510 that can be accessed by the project generation component 206 as needed. These visualizations 510 can be classified according to industry or industrial vertical (e.g., automotive, food and drug, oil and gas, pharmaceutical, etc.), type of industrial asset (e.g., a type of machine or industrial device), a type of industrial application (e.g., batch processing, flow control, web ten-

sion control, sheet metal stamping, water treatment, etc.), or other such categories. Predefined visualizations 510 can comprise visualizations in a variety of formats, including but not limited to HMI screens or windows, mashups that aggregate data from multiple pre-specified sources, AR overlays, VR objects representing 3D virtualizations of the associated industrial asset, or other such visualization formats. IDE system 202 can select a suitable visualization for a given object based on a predefined association between the object type and the visualization content.

[0039] In another example, markings applied to an engineering drawing by a user can be understood by some embodiments of the project generation component 206 to convey a specific design intention or parameter. For example, a marking in red pen can be understood to indicate a safety zone, two circles connected by a dashed line can be interpreted as a gearing relationship, and a bold line may indicate a camming relationship. In this way, a designer can sketch out design goals on an existing drawing in a manner that can be understood and leveraged by the IDE system 202 to generate code and visualizations. In another example, the project generation component 206 can learn permissives and interlocks (e.g., valves and their associated states) that serve as necessary preconditions for starting a machine based on analysis of the user's CAD drawings. Project generation component 206 can generate any suitable code (ladder logic, function blocks, etc.), device configurations, and visualizations based on analysis of these drawings and markings for incorporation into system project 302. In some embodiments, user interface component 204 can include design tools for developing engineering drawings within the IDE platform itself, and the project generation component 206 can generate this code as a background process as the user is creating the drawings for a new project. In some embodiments, project generation component 206 can also translate state machine drawings to a corresponding programming sequence, yielding at least skeletal code that can be enhanced by the developer with additional programming details as needed.

[0040] Also, or in addition, some embodiments of IDE system 202 can support goal-based automated programming. For example, the user interface component 204 can allow the user to specify production goals for an automation system being designed (e.g., specifying that a bottling plant being designed must be capable of producing at least 5000 bottles per second during normal operation) and any other relevant design constraints applied to the design project (e.g., budget limitations, available floor space, available control cabinet space, etc.). Based on this information, the project generation component 206 will generate portions of the system project 302 to satisfy the specified design goals and constraints. Portions of the system project 302 that can be generated in this manner can include, but are not limited to, device and equipment selections (e.g., definitions of how many pumps, controllers, stations, conveyors, drives, or other

assets will be needed to satisfy the specified goal), associated device configurations (e.g., tuning parameters, network settings, drive parameters, etc.), control coding, or HMI screens suitable for visualizing the automation system being designed.

[0041] Some embodiments of the project generation component 206 can also generate at least some of the project code for system project 302 based on knowledge of parts that have been ordered for the project being developed. This can involve accessing the customer's account information maintained by an equipment vendor to identify devices that have been purchased for the project. Based on this information the project generation component 206 can add appropriate automation objects 222 and associated code modules 508 corresponding to the purchased assets, thereby providing a starting point for project development.

[0042] Some embodiments of project generation component 206 can also monitor customer-specific design approaches for commonly programmed functions (e.g., pumping applications, batch processes, palletizing operations, etc.) and generate recommendations for design modules (e.g., code modules 508, visualizations 510, etc.) that the user may wish to incorporate into a current design project based on an inference of the designer's goals and learned approaches to achieving the goal. To this end, some embodiments of project generation component 206 can be configured to monitor design input 512 over time and, based on this monitoring, learn correlations between certain design actions (e.g., addition of certain code modules or snippets to design projects, selection of certain visualizations, etc.) and types of industrial assets, industrial sequences, or industrial processes being designed. Project generation component 206 can record these learned correlations and generate recommendations during subsequent project development sessions based on these correlations. For example, if project generation component 206 determines, based on analysis of design input 512, that a designer is currently developing a control project involving a type of industrial equipment that has been programmed and/or visualized in the past in a repeated, predictable manner, the project generation component 206 can instruct user interface component 204 to render recommended development steps or code modules 508 the designer may wish to incorporate into the system project 302 based on how this equipment was configured and/or programmed in the past.

[0043] In some embodiments, IDE system 202 can also store and implement guardrail templates 506 that define design guardrails intended to ensure the project's compliance with internal or external design standards. Based on design parameters defined by one or more selected guardrail templates 506, user interface component 204 can provide, as a subset of design feedback 518, dynamic recommendations or other types of feedback designed to guide the developer in a manner that ensures compliance of the system project 302 with inter-

nal or external requirements or standards (e.g., certifications such as TUV certification, in-house design standards, industry-specific or vertical-specific design standards, etc.). This feedback 518 can take the form of text-based recommendations (e.g., recommendations to rewrite an indicated portion of control code to comply with a defined programming standard), syntax highlighting, error highlighting, auto-completion of code snippets, or other such formats. In this way, IDE system 202 can customize design feedback 518 - including programming recommendations, recommendations of predefined code modules 508 or visualizations 510, error and syntax highlighting, etc. - in accordance with the type of industrial system being developed and any applicable in-house design standards.

[0044] Guardrail templates 506 can also be designed to maintain compliance with global best practices applicable to control programming or other aspects of project development. For example, user interface component 204 may generate and render an alert if a developer's control programming is deemed to be too complex as defined by criteria specified by one or more guardrail templates 506. Since different verticals (e.g., automotive, pharmaceutical, oil and gas, food and drug, marine, etc.) must adhere to different standards and certifications, the IDE system 202 can maintain a library of guardrail templates 506 for different internal and external standards and certifications, including customized user-specific guardrail templates 506. These guardrail templates 506 can be classified according to industrial vertical, type of industrial application, plant facility (in the case of custom in-house guardrail templates 506) or other such categories. During development, project generation component 206 can select and apply a subset of guardrail templates 506 determined to be relevant to the project currently being developed, based on a determination of such aspects as the industrial vertical to which the project relates, the type of industrial application being programmed (e.g., flow control, web tension control, a certain batch process, etc.), or other such aspects. Project generation component 206 can leverage guardrail templates 506 to implement rules-based programming, whereby programming feedback (a subset of design feedback 518) such as dynamic intelligent autocorrection, type-aheads, or coding suggestions are rendered based on encoded industry expertise and best practices (e.g., identifying inefficiencies in code being developed and recommending appropriate corrections).

[0045] Users can also run their own internal guardrail templates 506 against code provided by outside vendors (e.g., OEMs) to ensure that this code complies with in-house programming standards. In such scenarios, vendor-provided code can be submitted to the IDE system 202, and project generation component 206 can analyze this code in view of in-house coding standards specified by one or more custom guardrail templates 506. Based on results of this analysis, user interface component 204 can indicate portions of the vendor-provided code (e.g.,

using highlights, overlaid text, etc.) that do not conform to the programming standards set forth by the guardrail templates 506, and display suggestions for modifying the code in order to bring the code into compliance. As an alternative or in addition to recommending these modifications, some embodiments of project generation component 206 can be configured to automatically modify the code in accordance with the recommendations to bring the code into conformance.

[0046] In making coding suggestions as part of design feedback 518, project generation component 206 can invoke selected code modules 508 stored in a code module database (e.g., on memory 220). These code modules 508 comprise standardized coding segments for controlling common industrial tasks or applications (e.g., palletizing, flow control, web tension control, pick-and-place applications, conveyor control, etc.). In some embodiments, code modules 508 can be categorized according to one or more of an industrial vertical (e.g., automotive, food and drug, oil and gas, textiles, marine, pharmaceutical, etc.), an industrial application, or a type of machine or device to which the code module 508 is applicable. In some embodiments, project generation component 206 can infer a programmer's current programming task or design goal based on programmatic input being provided by a the programmer (as a subset of design input 512), and determine, based on this task or goal, whether one of the pre-defined code modules 508 may be appropriately added to the control program being developed to achieve the inferred task or goal. For example, project generation component 206 may infer, based on analysis of design input 512, that the programmer is currently developing control code for transferring material from a first tank to another tank, and in response, recommend inclusion of a predefined code module 508 comprising standardized or frequently utilized code for controlling the valves, pumps, or other assets necessary to achieve the material transfer.

[0047] Customized guardrail templates 506 can also be defined to capture nuances of a customer site that should be taken into consideration in the project design. For example, a guardrail template 506 could record the fact that the automation system being designed will be installed in a region where power outages are common, and will factor this consideration when generating design feedback 518; e.g., by recommending implementation of backup uninterruptable power supplies and suggesting how these should be incorporated, as well as recommending associated programming or control strategies that take these outages into account.

[0048] IDE system 202 can also use guardrail templates 506 to guide user selection of equipment or devices for a given design goal; e.g., based on the industrial vertical, type of control application (e.g., sheet metal stamping, die casting, palletization, conveyor control, web tension control, batch processing, etc.), budgetary constraints for the project, physical constraints at the installation site (e.g., available floor, wall or cabinet space;

dimensions of the installation space; etc.), equipment already existing at the site, etc. Some or all of these parameters and constraints can be provided as design input 512, and user interface component 204 can render the equipment recommendations as a subset of design feedback 518. In some embodiments, project generation component 206 can also determine whether some or all existing equipment can be repurposed for the new control system being designed. For example, if a new bottling line is to be added to a production area, there may be an opportunity to leverage existing equipment since some bottling lines already exist. The decision as to which devices and equipment can be reused will affect the design of the new control system. Accordingly, some of the design input 512 provided to the IDE system 202 can include specifics of the customer's existing systems within or near the installation site. In some embodiments, project generation component 206 can apply artificial intelligence (AI) or traditional analytic approaches to this information to determine whether existing equipment specified in design input 512 can be repurposed or leveraged. Based on results of this analysis, project generation component 206 can generate, as design feedback 518, a list of any new equipment that may need to be purchased based on these decisions.

[0049] In some embodiments, IDE system 202 can offer design recommendations based on an understanding of the physical environment within which the automation system being designed will be installed. To this end, information regarding the physical environment can be submitted to the IDE system 202 (as part of design input 512) in the form of 2D or 3D images or video of the plant environment. This environmental information can also be obtained from an existing digital twin of the plant, or by analysis of scanned environmental data obtained by a wearable AR appliance in some embodiments. Project generation component 206 can analyze this image, video, or digital twin data to identify physical elements within the installation area (e.g., walls, girders, safety fences, existing machines and devices, etc.) and physical relationships between these elements. This can include ascertaining distances between machines, lengths of piping runs, locations and distances of wiring harnesses or cable trays, etc. Based on results of this analysis, project generation component 206 can add context to schematics generated as part of system project 302, generate recommendations regarding optimal locations for devices or machines (e.g., recommending a minimum separation between power and data cables), or make other refinements to the system project 302. At least some of this design data can be generated based on physics-based rules 516, which can be referenced by project generation component 206 to determine such physical design specifications as minimum safe distances from hazardous equipment (which may also factor into determining suitable locations for installation of safety devices relative to this equipment, given expected human or vehicle reaction times defined by the physics-based rules 516),

material selections capable of withstanding expected loads, piping configurations and tuning for a specified flow control application, wiring gauges suitable for an expected electrical load, minimum distances between signal wiring and electromagnetic field (EMF) sources to ensure negligible electrical interference on data signals, or other such design features that are dependent on physical rules.

[0050] In an example use case, relative locations of machines and devices specified by physical environment information submitted to the IDE system 202 can be used by the project generation component 206 to generate design data for an industrial safety system. For example, project generation component 206 can analyze distance measurements between safety equipment and hazardous machines and, based on these measurements, determine suitable placements and configurations of safety devices and associated safety controllers that ensure the machine will shut down within a sufficient safety reaction time to prevent injury (e.g., in the event that a person runs through a light curtain).

[0051] In some embodiments, project generation component 206 can also analyze photographic or video data of an existing machine to determine inline mechanical properties such as gearing or camming and factor this information into one or more guardrail templates 506 or design recommendations.

[0052] As noted above, the system project 302 generated by IDE system 202 for a given automaton system being designed can be built upon an object-based architecture that uses automation objects 222 as building blocks. FIG. 6 is a diagram illustrating an example system project 302 that incorporates automation objects 222 into the project model. In this example, various automation objects 222 representing analogous industrial devices, systems, or assets of an automation system (e.g., a process, tanks, valves, pumps, etc.) have been incorporated into system project 302 as elements of a larger project data model 602. The project data model 602 also defines hierarchical relationships between these automation objects 222. According to an example relationship, a process automation object representing a batch process may be defined as a parent object to a number of child objects representing devices and equipment that carry out the process, such as tanks, pumps, and valves. Each automation object 222 has associated therewith object properties or attributes specific to its corresponding industrial asset (e.g., those discussed above in connection with FIG. 4), including executable control programming for controlling the asset (or for coordinating the actions of the asset with other industrial assets) and visualizations that can be used to render relevant information about the asset during runtime.

[0053] At least some of the attributes of each automation object 222 are default properties defined by the IDE system 202 based on encoded industry expertise pertaining to the asset represented by the objects. Other properties can be modified or added by the developer as

needed (via design input 512) to customize the object 222 for the particular asset and/or industrial application for which the system projects 302 is being developed. This can include, for example, associating customized control code, HMI screens, AR presentations, or help files associated with selected automation objects 222. In this way, automation objects 222 can be created and augmented as needed during design for consumption or execution by target control devices during runtime.

[0054] Once development on a system project 302 has been completed, commissioning tools supported by the IDE system 202 can simplify the process of commissioning the project in the field. When the system project 302 for a given automation system has been completed, the system project 302 can be deployed to one or more target control devices for execution. FIG. 7 is a diagram illustrating commissioning of a system project 302. Project deployment component 208 can compile or otherwise translate a completed system project 302 into one or more executable files or configuration files that can be stored and executed on respective target industrial devices of the automation system (e.g., industrial controllers 118, HMI terminals 114 or other types of visualization systems, motor drives 710, telemetry devices, vision systems, safety relays, etc.).

[0055] Conventional control program development platforms require the developer to specify the type of industrial controller (e.g., the controller's model number) on which the control program will run prior to development, thereby binding the control programming to a specified controller. Controller-specific guardrails are then enforced during program development which limit how the program is developed given the capabilities of the selected controller. By contrast, some embodiments of the IDE system 202 can abstract project development from the specific controller type, allowing the designer to develop the system project 302 as a logical representation of the automation system in a manner that is agnostic to where and how the various control aspects of system project 302 will run. Once project development is complete and system project 302 is ready for commissioning, the user can specify (via user interface component 204) target devices on which respective aspects of the system project 302 are to be executed. In response, an allocation engine of the project deployment component 208 will translate aspects of the system project 302 to respective executable files formatted for storage and execution on their respective target devices.

[0056] For example, system project 302 may include - among other project aspects - control code, visualization screen definitions, and motor drive parameter definitions. Upon completion of project development, a user can identify which target devices- including an industrial controller 118, an HMI terminal 114, and a motor drive 710 - are to execute or receive these respective aspects of the system project 302. Project deployment component 208 can then translate the controller code defined by the system project 302 to a control program file 702 formatted

for execution on the specified industrial controller 118 and send this control program file 702 to the controller 118 (e.g., via plant network 116). Similarly, project deployment component 208 can translate the visualization definitions and motor drive parameter definitions to a visualization application 704 and a device configuration file 708, respectively, and deploy these files to their respective target devices for execution and/or device configuration.

[0057] In general, project deployment component 208 performs any conversions necessary to allow aspects of system project 302 to execute on the specified devices. Any inherent relationships, handshakes, or data sharing defined in the system project 302 are maintained regardless of how the various elements of the system project 302 are distributed. In this way, embodiments of the IDE system 202 can decouple the project from how and where the project is to be run. This also allows the same system project 302 to be commissioned at different plant facilities having different sets of control equipment. That is, some embodiments of the IDE system 202 can allocate project code to different target devices as a function of the particular devices found on-site. IDE system 202 can also allow some portions of the project file to be commissioned as an emulator or on a cloud-based controller.

[0058] As an alternative to having the user specify the target control devices to which the system project 302 is to be deployed, some embodiments of IDE system 202 can actively connect to the plant network 116 and discover available devices, ascertain the control hardware architecture present on the plant floor, infer appropriate target devices for respective executable aspects of system project 302, and deploy the system project 302 to these selected target devices. As part of this commissioning process, IDE system 202 can also connect to remote knowledgebases (e.g., web-based or cloud-based knowledgebases) to determine which discovered devices are out of date or require firmware upgrade to properly execute the system project 302. In this way, the IDE system 202 can serve as a link between device vendors and a customer's plant ecosystem via a trusted connection in the cloud.

[0059] Copies of system project 302 can be propagated to multiple plant facilities having varying equipment configurations using smart propagation, whereby the project deployment component 208 intelligently associates project components with the correct industrial asset or control device even if the equipment on-site does not perfectly match the defined target (e.g., if different pump types are found at different sites). For target devices that do not perfectly match the expected asset, project deployment component 208 can calculate the estimated impact of running the system project 302 on non-optimal target equipment and generate warnings or recommendations for mitigating expected deviations from optimal project execution.

[0060] As noted above, some embodiments of IDE system 202 can be embodied on a cloud platform. FIG. 8 is

a diagram illustrating an example architecture in which cloud-based IDE services 802 are used to develop and deploy industrial applications to a plant environment. In this example, the industrial environment includes one or more industrial controllers 118, HMI terminals 114, motor drives 710, servers 801 running higher level applications (e.g., ERP, MES, etc.), and other such industrial assets. These industrial assets are connected to a plant network 116 (e.g., a common industrial protocol network, an Ethernet/IP network, etc.) that facilitates data exchange between industrial devices on the plant floor. Plant network 116 may be a wired or a wireless network. In the illustrated example, the high-level servers 810 reside on a separate office network 108 that is connected to the plant network 116 (e.g., through a router 808 or other network infrastructure device).

[0061] In this example, IDE system 202 resides on a cloud platform 806 and executes as a set of cloud-based IDE service 802 that are accessible to authorized remote client devices 504. Cloud platform 806 can be any infrastructure that allows shared computing services (such as IDE services 802) to be accessed and utilized by cloud-capable devices. Cloud platform 806 can be a public cloud accessible via the Internet by devices 504 having Internet connectivity and appropriate authorizations to utilize the IDE services 802. In some scenarios, cloud platform 806 can be provided by a cloud provider as a platform-as-a-service (PaaS), and the IDE services 802 can reside and execute on the cloud platform 806 as a cloud-based service. In some such configurations, access to the cloud platform 806 and associated IDE services 802 can be provided to customers as a subscription service by an owner of the IDE services 802. Alternatively, cloud platform 806 can be a private cloud operated internally by the industrial enterprise (the owner of the plant facility). An example private cloud platform can comprise a set of servers hosting the IDE services 802 and residing on a corporate network protected by a firewall.

[0062] Cloud-based implementations of IDE system 202 can facilitate collaborative development by multiple remote developers who are authorized to access the IDE services 802. When a system project 302 is ready for deployment, the project 302 can be commissioned to the plant facility via a secure connection between the office network 108 or the plant network 116 and the cloud platform 806. As discussed above, the industrial IDE services 802 can translate system project 302 to one or more appropriate executable files - control program files 702, visualization applications 704, device configuration files 708, system configuration files 812 - and deploy these files to the appropriate devices in the plant facility to facilitate implementation of the automation project.

[0063] FIG. 9 is an example development interface 902 that can be rendered by one or more embodiments of the industrial IDE system's user interface component 204. Development interface 902 is organized into panels and workspaces in a manner to be described in more

detail herein, and supports automated and manual curation features that declutter the development space and bring a subset of project editing functions that are relevant to a current development task into focus. These features can improve the user's development workflow experience by filtering out selectable options that are not relevant to a current development task, allowing relevant editing tools and information to be located more easily.

[0064] The basic structure of development interface 902 comprises a canvas area 930 in which resides a workspace canvas 940 (having an associated tab 932), a global panel control bar 920 on the right-side edge of the interface 902 (to the right of the canvas area 930), a menu bar 904 along the top edge of the interface 902, and a tool bar 906 below the menu bar 904. Other panels can be selectively added or removed from the interface's workspace using visibility control icons on the global panel control bar 920 or via selectable options under the View option of the menu bar 904. These panels can be added to or removed from three main panel area - a left global panel area 922, a bottom global panel area 924, and a right global panel area 928. In the example scenario depicted in FIG. 9, a Properties panel 936 is visible in the right global panel area 928, and an Explorer panel 910 and a Toolbox panel 912 have been rendered in a vertically stacked arrangement in the left global panel area 922. Development interface 902 can also include a search bar 934 for searching the open project using text string searches. The search bar 934 can also be used for inserting text or initiating a shortcut in some embodiments.

[0065] FIG. 10a is a close-up view of the global panel control bar 920 illustrating an example organization of panel visibility icons. Visibility icons are organized vertically into three groups along the global panel control bar 920, the respective groups residing in a global left panel control area 914, a global right panel control area 916, and a global bottom panel control area 918 of the control bar 920. The three panel control areas are labeled with respective header icons 1002, 1004, and 1006 illustrating which global panel area (left, right, or bottom) are controlled by the associated icons. In the illustrated example, the left panel control area 914 comprises an Explorer visibility icon 1008 that, in response to selection, toggles the visibility of the Explorer panel 910 in the left global panel area 922. The right panel control area 916 comprises three visibility icons 1010a-1010c, which control visibility of a Properties panel (visibility icon 1010a), an Online panel (visibility icon 1010b), and a Cross Reference panel (visibility icon 1010c), respectively, in the right global panel area 928. The bottom panel control area 918 comprises two visibility icons 1012a and 1012b, which control visibility of an Errors panel (visibility icon 1012a) and an Output panel (visibility icon 1012b), respectively, in the bottom global panel area 924.

[0066] The visibility icons on global panel control bar 920 can act as toggle buttons that toggle the visibility of their corresponding panels, such that selecting the icon

a first time causes the corresponding panel to be rendered in its designated area, and selecting the icon a second time removes its corresponding panel from its designated area. The visibility icons can be color animated such that the color of the icon indicates the visible or hidden state of the corresponding panel (e.g., black for hidden and blue for visible).

[0067] FIG. 10b is an example View menu 1014 that can be rendered as a drop-down menu in response to selection of the View option in the menu bar 904. View menu 1014 renders selectable visibility controls corresponding to, and having the same functionality as, the visibility icons rendered on the global panel control bar 920, allowing the user to selectively render and hide panels using either this menu 1014 or the global panel control bar 920. Similar to the global panel control bar 920, the selectable visibility controls are organized according to Left Panels, Right Panels, and Bottom Panels. Unlike the global panel control bar 920, the selectable controls of the View menu 1014 are rendered as selectable text rather than icons, with checkmarks indicating panels that are currently visible.

[0068] In some embodiments, any panels associated with a global panel area (left, right, or bottom) that have been set to be pinned (to be discussed below) can be rendered visible or invisible with a single selection by selecting either the header icon (icon 1002, 1004, or 1006) corresponding to that area in the global panel control bar 920 or the header text for that set of panels (e.g., the Right Panels header 1016) in the View menu 1014.

[0069] In some embodiments, the panels whose visibility is controlled from the global panel control bar 920 can be global panels that are relevant to all development tasks or contexts supported by the industrial IDE system 202 (content panels, which are relevant to specific development tasks or contexts, will be described below). In the example depicted in FIG. 10a and 10b, the global panels include an Explorer panel through which a user can browse and select aspects or elements of the automation project, a Properties panel that renders property information for a selected element within canvas area 930, an Online panel that renders communication statistics for the industrial IDE system, a Cross Reference panel that renders cross reference information for a selected element within canvas area 930 (e.g., by listing all usages or instances of the selected element within the industrial automation system project), an Output panel that renders output states, and an Errors panel that lists active and/or historical development or runtime errors. However, any type of global panel can be supported by the development interface 902 without departing from the scope of one or more embodiments. For example, a Toolbox panel that renders a set of global editing tools - or links to a specific subset of editing tools of selected categories - may also be supported as a global panel.

[0070] In some embodiments, a panel's transition between visible and invisible states can be animated, such that invoking a panel causes the panel to slide from a

designated edge of the development interface 902 (left, right or bottom), toward the middle of the interface 902 until the panel is fully extended and visible. Similarly, instructing a visible panel to switch to the hidden state causes the panel to retract toward the edge from which the panel initially extended.

[0071] Panels supported by the IDE system 202 can be generally classified into two types - global panels and content panels. Global panels are globally applicable to all development contexts, and can include, but are not limited to, the global panels discussed above. The visibility icons corresponding to global panels are always fixed on the panel control bar 920.

[0072] In contrast to global panels, content panels are not globally applicable, but rather are relevant or applicable only to a specific development task or context (e.g., ladder logic control programming, function block diagram control programming, sequential function chart control programming, structured text control programming, HMI screen development, device configuration, controller tag definition, etc.). Content panels can include, but are not limited to, a Layers panel that facilitates browsing through layers of graphical content (e.g., engineering drawings, HMI screens, etc.), an Alarms panel that renders configurable alarm definition data for selected alarm tags, a Logic Editor panel that renders selectable program elements that can be added to a ladder logic program (e.g., output coils, contacts, function blocks, etc.), an HMI screen development panel that renders selectable graphical elements that can be added to an HMI screen, or other such content panels. Visibility icons for content panels are located on the canvas toolbar 938 (see, e.g., FIG. 9) along the top edge of the canvas 940, and the set of content panel visibility icons available on the toolbar 938 is a function of the type of content (e.g., control programming, HMI development screens, etc.) rendered in the canvas 940. Thus, content panels will only be available for selection if the user is currently focused on the development task or context to which the content panel is relevant (based on which canvas 940 currently has focus within the development interface 902, and the type of project content rendered by the canvas 940). Example types of project content that can be associated with a dedicated set of content panels (and associated visibility icons) can include, but are not limited to, a ladder logic routine, a function block diagram routine, a structured text routine, a sequential function chart routine, a tag database, an HMI screen or application, a faceplate, various types of device views (e.g., controllers, drives, I/O modules, etc.), an engineering drawing, or other such content types.

[0073] In general, any of the panels associated with the left global panel area 922, right global panel area 928, or bottom global panel area 924 can be selectively set to be a pinned panel or an overlay panel. FIG. 11a is a view of the top right corner of development interface 902 depicting a Properties panel 936 pinned in the right global panel area 928. Visibility icon 1010a - correspond-

ing to the Properties panel 936 - is highlighted to indicate that the Properties panel 936 is visible. Any of the panels can be selectively set to be pinned or unpinned (i.e. overlaid) by selecting a suitable control; e.g., a control selected from a drop-down panel setting menu that can be invoked by selecting the panel menu icon 1102 in the top right corner of the panel. In some embodiments, a panel can also be selectively rendered as a pinned panel or as an overlay panel by selecting an appropriate control from a right-click menu associated with the corresponding visibility icon in the global panel control bar 920. Setting a panel to be pinned simulates pinning the panel to the background while visible, while setting a panel to be an overlay (unpinned) causes the panel to be rendered as an overlay over any pinned panels, or other interface content (e.g., canvas content), that may already be invoked in that part of the display.

[0074] When a pinned panel is invoked, user interface component 204 reduces the width of the canvas area 930 (or reduces the canvas area's height in the case of pinned panels in the bottom global panel area 924) to accommodate the pinned panel. This also causes one or more canvases 940 within the canvas area 930 to be similarly reduced in size. This can be seen in FIG. 11a, where the right edge 1112 of the canvas area 930 has shifted toward the middle of the interface 902 to accommodate the width of the pinned panel 936, such that the right edge 1112 of the canvas area 930 is abutted against the left edge of the panel 936. When an overlay panel is invoked, the size of the canvas area 930 is not adjusted, and instead the panel is rendered as an overlay over a portion of the canvas, obscuring a portion of the canvas content behind the panel.

[0075] FIG. 11b is a view of the top right corner of the development interface 902 depicting selection of an Online panel 1104 as an overlaid panel in the right global panel area 928. As shown in this figure, selection of the Online panel visibility icon 1010b while the pinned Properties panel 936 is visible causes the Online panel 1104 - which is currently set to be an overlay panel - to be displayed over the Properties panel. A panel set to be an overlay can be rendered with a shadow effect 1106 to convey that the panel is an overlay rather than a pinned panel (which is not rendered with a shadow effect). The width of the overlaid panel (e.g., Online panel 1104 in FIG. 11b) can be resized by clicking on or otherwise selecting the outer edge of the panel and sliding the edge inward or outward. Reducing the width of the overlay panel causes portions of any pinned panels underneath the overlay panel to be revealed. Although pinned and overlay panel effects are illustrated in FIGs. 11a and 11b with reference to the right global panel area 928, these effects are also applicable to the left global panel area 922 and bottom global panel area 924.

[0076] FIG. 11c is a view of the top right corner of development interface 902 depicting two pinned panels - Properties panel 936 and Cross Reference Panel 1108 - that are visible simultaneously. In this example, the

Properties panel visibility icon 1010a and the Cross Reference panel visibility icon 1010b have been toggled on. Since both of these panels are currently set to be pinned panels, both panels 936 and 1108 are visible, stacked vertically in the right global panel area. In an example embodiment, if only one pinned panel is selected to be visible in a given area, that panel can be sized vertically to encompass the entire height of the panel area (e.g., right global panel area 928). If a second pinned panel is invoked, the two panels will be sized vertically such that both panels will fit within the panel area in a vertically stacked arrangement. The horizontal sizes of the stacked pinned panels can be changed by clicking and dragging the horizontal interface 1110 between the two panels upward or downward (where an upward drag decreases the size of the upper panel and increases the size of the lower panel, while a downward drag performs the reverse resizing).

[0077] In some scenarios, an overlaid panel may be sized or oriented to allow a portion of a pinned panel behind the overlaid panel to remain visible. FIG. 11d is a view of the top right corner of development interface 902 in which a Toolbox panel 1114 is rendered as an overlay above Properties panel 936. However, the top of Toolbox panel 1114 is below the top of Properties panel 936, allowing a portion of the Properties panel 936 to remain visible. FIG. 11e depicts a scenario in which the Toolbox panel 1114 of FIG. 11d is switched to be a pinned panel, thereby causing panels 936 and 1114 to be stacked vertically.

[0078] As noted above, a panel can be set to be pinned by selecting a control associated with the panel. In some embodiments, a panel can also be pinned to a global panel area using a drag-and-drop action. FIG. 12 is a view of the top right corner of development interface 902 depicting a panel drop area 1202 for the right global panel area 928 according to such embodiments. According to an example embodiment, if no panels associated with the right global panel area 928 are set to be pinned (that is, the three available panels for the right global panel area 928 are currently set to be overlays, such that invoking the panel causes the panel to be rendered in the right global panel area 928 as an overlay), selecting the header icon 1004 for the right global panel area 928 causes an empty panel drop area 1202 to be rendered in the right global panel area 928. Any of the three panels available for the right global panel area 928 can be set to be pinned panels by dragging the corresponding visibility icon 1010 for the panel to the panel drop area 1202, as indicated by the arrow in FIG. 12. Pinned panels can also be unpinned (that is, set to be overlay panels) by dragging the panels from the drop area 1202 back to the global panel control bar 920. This drag-and-drop approach can be used to pin panels to any of the three global panel areas (left, right, and bottom).

[0079] In some embodiments, pinned visible panels can also be selectively collapsed or expanded. FIG. 13a depicts two horizontally stacked pinned panels (a Prop-

erties panel 936 and an Allocation panel 1302) in a default non-collapsed state. In this state, the content windows of both panels are visible below the respective header bars 1304 and 1306. A panel can be collapsed by selecting the header bar 1304 or 1306 corresponding to that panel. FIG. 13b depicts the Allocation panel 1302 in the collapsed state as a result of clicking on or otherwise selecting the header bar 1306 for that panel. When the lower panel - the Allocation panel 1302 in this case - is collapsed, the content window for that panel is rendered invisible and the header bar 1306 moves to the bottom of the panel area, while the content window for the upper panel (the Properties panel 936 in this case) is lengthened to fill the remaining panel area space, exposing more of that window's content. FIG. 13c depicts the Properties panel 936 collapsed as a result of clicking on or otherwise selecting the header bar 1304 for that panel. When the upper panel is collapsed, the content window for that panel is rendered invisible, and the header bar 1306 for the lower panel moves upward to a location just below the header bar 1304 of the upper panel. The content window of the lower panel fills the remaining panel area space, revealing more of the content of that panel.

[0080] Returning briefly to FIG. 9, the canvas area 930 is the primary work area for the IDE system's development interface 902, and is bounded by the left global panel area 922, the right global panel area 928, the bottom global panel area 924, and the menu bar 904. In general, the canvas area 930 contains the one or more workspace canvases 940 on which the user interface component 204 renders components of the system project, such as ladder logic or other types of control code, program routines, controller tag definitions, development views of visualization screens, device configurations, engineering drawings, or other project components. The canvas area 930 is also the space with which the user interacts with these components - leveraging editing tools and information provided by the global and content panels - to perform such development functions as developing controller code (e.g., ladder logic, function block diagrams, structured text, etc.), developing visualizations for the automation system (e.g., HMI screens, AR/VR presentations, mashups, etc.), configuring device parameter settings, defining controller tags, developing engineering drawings, or other such project development functions.

[0081] FIG. 14 is a closer view of an example canvas 940 within the canvas area 930. Each canvas 940 within the canvas area 930 can be associated with at tab 932, selection of which brings the corresponding canvas 940 into focus. Canvas 940 can also have an associated toolbar 938 comprising selectable icons and/or fields that allows the user to set properties for the associated canvas 940, such as zoom levels, view formats, grid line visibility, or other such properties. In the example depicted in FIG. 14, the canvas's toolbar 938 is located below tab 932.

[0082] In some embodiments, the canvas's toolbar 938

can also contain visibility icons for any content panels associated with the type of content (e.g., ladder logic, function block diagram, structured text, HMI screens in development, device parameters, engineering drawings, etc.) currently being rendered in the canvas 940. Similar to the global panel visibility icons located on the global panel control bar 920, selection of a content panel visibility icon from a canvas's toolbar 938 toggles the visibility of the panel associated with the selected icon. In some embodiments, when a content panel is made visible, the content panel can be rendered at a predefined designated location either in one of the global panel areas or adjacent to one of the global panel areas. Content panels may also be moved to a selected location within the interface workspace in some embodiments. Similar to global panels, content panels can be selectively set to be either pinned or overlaid.

[0083] Although the illustrated example depicts panel visibility icons as being rendered in the canvas's toolbar 938, panel visibility icons can also be rendered elsewhere on the development interface 902 in some embodiments; e.g., on the main tool bar 906 below the menu bar 904. In such embodiments, the list of panel visibility icons rendered in this space at a given time will be a function of the type of project content that currently has focus (e.g., the content of the particular canvas 940 that currently has focus). In other embodiments, user interface component 204 may add available content panel visibility icons to the global panel control bar 920 in their own designated grouping, based on the type of project content or development task currently being performed.

[0084] Canvas area 930 can comprise one or more tabbed canvases 940, with each canvas 940 associated with a tab 932. User interface component 204 allows the user to establish as many tabbed canvases 940 within the canvas area 930 as desired, with each tab 932 rendering a different aspect of the automation system project. Multiple tabbed canvases 940 can be stacked in the canvas area 930 either horizontally or vertically. FIG. 15 is a view of development interface 902 in which two canvases 940a and 940b have been stacked horizontally. Stacking tabs in this manner - either horizontally or vertically - allows content of both canvases 940a and 940b to be rendered simultaneously.

[0085] Users may also select to render multiple canvases 940 as overlays on top of one another. FIGs. 16a and 16b are views of two overlaid canvases 940a and 940b. In this example scenario, the first canvas 940a is rendering a ladder logic routine being developed for an industrial controller, and the second canvas 940b is rendering a tag database for the controller. FIG. 16a depicts a scenario in which tab 932a is selected, causing the corresponding ladder logic canvas 940a to be rendered in the canvas area 930. FIG. 16b depicts a scenario in which tab 932b is selected, causing the corresponding tag database canvas 940b to be rendered in the canvas area 930.

[0086] In the aggregate, the basic layout of the devel-

opment interface 902 together with the panel control and tab manipulation functionalities described above can offer the user a fluid development workspace that affords a great deal of control over the balance between usable workspace and editing function availability. Moreover, since the user interface component 204 dynamically filters the available editing tools according to the user's current development task or focus - by making only a subset of content panels that are relevant to the current task available for selection - the development interface 902 substantially declutters the development workspace by removing panels and editing functions that are not relevant to the task at hand.

[0087] FIGs. 17a-17e are views of various example layouts of the IDE system's development interface 902, illustrating increasing degrees of IDE content density that can be supported by the interface 902. FIG. 17a is a view of interface 902 in which a single canvas 940a is open and no left, right, or bottom panels are invoked. This substantially maximizes the size of the canvas 940 since no development workspace is being consumed by global or content panels, thereby displaying a substantially maximized amount of canvas content (e.g., control programming, tag database information, etc.). The panel control bar 920 remains pinned to the right-side edge of the development interface 902 to allow the user to invoke panels as needed. As noted above, in addition to the global panel visibility icons, the panel control bar 920 will render a relevant subset of visibility icons corresponding to content panels that are relevant to the task being performed in the active canvas 940a (e.g., ladder logic programming, FBD programming, structured text programming, HMI screen development, device configuration, network configuration, etc.).

[0088] FIG. 17b is a view of interface 902 in which an Explorer panel 910 has been rendered visible in the left global panel area 922 and a Properties panel 936 has been rendered in the right global panel area 928. These panels can be rendered visible using any of the techniques described above (e.g., selection from the panel control bar 920 or from the View menu option). Both panels 910 and 936 are set to be pinned, and so the canvas 940a has been reduced in width to accommodate the panels 910 and 922 so that none of the canvas content 940a is obscured by the panels.

[0089] FIG. 17c is a view of development interface 902 in which a Layers panel 1702 (a content panel specific to the particular task being performed in the canvas 940a) has been added to the previous view. The Layers panel 1702 has been added as an overlay panel to the left of the Properties panel 936, and so will obscure a portion of the canvas content corresponding to that space. FIG. 17d adds further content to the previous view by adding a second canvas 940b, which is stacked horizontally with the original canvas 940a. The user can select which canvas 940 has the current focus by selecting the tab 932a or 932b corresponding to the desired canvas 940. This configuration allows the user to view content of both can-

vases 940 simultaneously (e.g., a control program and a tag database, a control program and a device view, etc.) while also affording the user access to the editing tools, information, and navigation structures associated with the Explorer panel 910, Properties panel 936, and Layers panel 1702.

[0090] FIG. 17e is a view of development interface in which a third canvas 940c is added to the previous view, stacked vertically with the two previous canvases 940a and 940b. As illustrated in this figure, canvases 940 can be selectively stacked either horizontally or vertically, or both horizontally and vertically, within the canvas area 930.

[0091] As illustrated by the examples depicted in FIGs 17a-17e, the development interface's layout and customization features grant the user considerable flexibility with regard to customizing or curating canvas layouts and behaviors, as well as selective rendering of project data and editing tools. Moreover, editing tools and views available to the user at a given time are intelligently curated by the user interface component 204 as a function of the user's current development task or context, which may be determined based on the identity of the canvas 940 that currently has focus and the content of that canvas 940. For example, if the user selects a canvas 940 in which a structured text program is being developed, only a subset of the interface's total library of content panels that are relevant to structured text program development will be made available to the user (e.g., by adding visibility icons corresponding to those panels to the panel control bar 920).

[0092] Some of the global and content panels supported by some embodiments of the development interface will now be discussed. FIG. 18 is a view of the Explorer panel 910, which resides in the left global panel 922 area when invoked. Explorer panel 910 serves as a means for navigating and viewing content of a system project, and supports numerous ways for performing this navigation. The Explorer panel 910 itself supports a number of different viewing categories, which are represented by selectable explorer icons 1806 rendered on an explorer view control bar 908 pinned to the left-side edge of the Explorer panel 910. Selection of an explorer icon 1806 determines one or both of the type of project content to be browsed via the Explorer panel 910 or a format in which the browsable project content is rendered on the Explorer panel 910.

[0093] Explorer panel 910 also comprises a panel header 1802, the text of which identifies the set of explorer tools that are currently visible (e.g., "System" in FIG. 18). For explorer views that offer a choice of alternative presentation formats of the content represented by the explorer icon 1806, horizontally stacked tabs 1804a and 1804b are located below the panel header 1802 for selecting from among the available views. Below the tabs 1802a and 1804b (or below the header 1802 if the current Explorer tool set has only one view) is the Explorer panel content area 1808 in which the currently

selected explorer tools are rendered. As will be discussed and illustrated below, the content rendered in the content area 1808 is a function of the explorer icon 1806 currently selected as well as the tab 1804 that currently has focus. For example, the selected explorer icon 1806 can determine the browsable project content to be rendered in the Explorer panel 910, and the selected tab 1804 determines a presentation format or organization of this browsable project content. For some views supported by Explorer panel 910, selection of an explorer icon 1806 may set a category of content to be rendered in the content area 1808, while selection of a tab can set the particular sub-category of rendered content within the main category.

[0094] FIGs. 19a-19b are view of the Explorer panel 910 in isolation, with the System view currently selected. The Explorer panel's System view can be invoked by selecting the System icon 1904 in the explorer view control bar 908. The System view offers two tabbed views - Logical (tab 1804a) and Execution (tab 1804b). FIG. 19a depicts the Logical System view rendered in response to selection of the Logical tab 1804a. The Logical System view renders a Logical System navigation tree 1902 in the content area 1808 comprising selectable nodes organized hierarchically. Selection of one of the nodes of the navigation tree 1902 associated with viewable project content causes content corresponding to the selected node to be rendered in the canvas 940 that currently has focus, or causes an appropriate panel to be rendered on the development interface 902 for display of the content (depending on the node selected and the corresponding content).

[0095] Project aspects that can be selected via the Logical System navigation tree 1902 can include, but are not limited to, control programs or routines (e.g., the RLL_01 and ST_01 nodes, which are listed in FIG. 19a under the Prog1 and Prog2 parent nodes, respectively, in FIG. 19), tags and/or parameters associated with a program (e.g., Tags/Params nodes in FIG. 19a, which are also listed under the parent nodes of their corresponding control programs), visualizations, alarm configurations, device configurations or parameter settings, trends, security settings, test results, or other such project aspects. In general, the nodes rendered in the Logical System navigation tree 1902 reflect elements that exist for the present automation system project.

[0096] In general, the Logical System view organizes system elements according to processes, production areas, or plant facilities within an industrial enterprise. FIG. 20 is an example Explorer panel 910 depicting a Logical System navigation tree 1902 for an example automation system project. As shown in this example, the Logical System navigation tree 1902 can organize aspects of the project hierarchically. The user can define parent nodes 2002 representing different processes, production areas, or plant facilities within the industrial enterprise (e.g., Extraction, Fermentation, Distillation, etc.). Sub-nodes 2004 can also be defined as child nodes of the parent

nodes 2002 if the process, production area, or plant facility is to be further broken down into sections (e.g., LIC551, P561, PIC535, etc.).

[0097] Below one or more of these user-defined nodes are selectable nodes representing aspects of the parent node that can be viewed and configured by the user. These can include logic nodes 2006 representing control programming associated with the parent node, visualization nodes 2008 representing HMI applications or other types of visualization applications associated with the parent node, tags and parameter nodes 2010 representing tags and device parameters defined or configured for the parent node, device nodes (not shown in FIG. 20) representing devices associated with the parent node (e.g., industrial controllers, motor drives, etc.) or other such system project components. In general, the path through tree 1902 to a node represents a logical path to the corresponding project aspect, defined in terms of the user's plant layout or process layout.

[0098] FIG. 19b is a view of the Explorer panel 910, in which the Execution System view rendered in response to selection of the Execution tab 1804b. This view renders similar content to that of the Logical System view described above in connection with FIGs. 19a and 20, but organized in a hierarchical Execution System navigation tree 1906 according to the execution devices (e.g., industrial controllers) on which the various aspects of the automation system reside and execute. This differs from the plant-based organization offered by Logical system navigation tree 1902. The path through tree 1906 to a node represents an execution path to the corresponding project aspect.

[0099] In some embodiments, the manner in which a user interacts with a node of the System navigation tree will determine how the content associated with the selected node is presented. FIG. 21a illustrates an example response of the user interface component 204 when a user selects, but does not launch, a ladder logic node 2102 representing a ladder logic program of the system project (RLL_01). The node 2102 can be selected, for example, by performing a single mouse click on the node 2102 such that the node is highlighted. When the node 2102 is selected in this manner, information about the selected ladder logic program will be rendered in the Properties panel 936 (if the Properties panel 936 is currently visible).

[0100] FIG. 21b illustrates an example response of the user interface component 204 when a user launches the ladder logic node 2102; e.g., by double-clicking on the node 2102. When a node in the System navigation tree 1902 or 1906 is double-clicked or otherwise instructed to launch, content or workspace associated with the node 2102 is rendered on a tabbed canvas 940. Double-clicking on the node 2102 can cause a new canvas 940 to be opened in the canvas area 930, or may cause a canvas 940 that currently has focus to render the content associated with the node 2102.

[0101] FIG. 21c illustrates an example response of the

user interface component 204 when a user right-clicks on the node 2102. Right-clicking on a node of the System navigation tree 1902 can cause a context menu 2104 to be rendered near the node 2102. Context menu 2104 renders a list of selectable options that are specific to the type of node selected. For example, if the selected node represents an industrial controller, context menu 2104 may list options to add an I/O module to the controller, to add a device to the controller (e.g., a drive), or options for other controller-specific configuration actions. The context menu 2104 may also include options for configuring the System navigation tree 1902 itself, such as copying, pasting, and deleting nodes.

[0102] FIGs. 22a and 22b are views of the Explorer panel 910 with the Application view currently selected. The Application view is invoked by selecting the Application icon 2202 in the explorer view control bar 908. The Application view lists applications (e.g., controller programs, HMI applications) that make up the automation system project in a browsable format. In this example, the Application view allows users to view controller application information by selecting the Controller tab 1804a, and to view HMI application information by selecting an HMI tab 1804b.

[0103] Selecting the Controller tab 1804a renders a Controller navigation tree 2204 in the Explorer panel content area 1808. The Controller navigation tree 2204 comprises nodes representing controller tags, controller parameters, control programming (e.g., ladder logic, structured text, function block diagram, etc.), handler routines (e.g., fault handlers, power-up handlers, etc.), and other such aspects of industrial controllers that make up the automation system project. These nodes are organized in the Controller navigation tree 2204 according to the controller with which the nodes are associated. Selection of a controller application node can render property information for the selected controller application in the Properties panel 936 (e.g. via single-click interaction) or can render the code for the selected application in a canvas 940 (e.g., via double-click interaction). FIG. 23 is a view of a canvas 940 on which a portion of an example structure text program is rendered in response to selection of a structured text application node from the Controller navigation tree 2204 or the System navigation tree 1902. FIG. 24 is a view of a canvas 940 on which a portion of an example function block diagram program is rendered in response to selection of a function block diagram application node from the Controller navigation tree 2204 or the System navigation tree 1902.

[0104] Similarly, selecting the HMI tab 1804b renders an HMI navigation tree 2206 in the Explorer panel content area 1808. This tree 2206 lists any HMI projects (or other types of visualization projects) associated with the automation system project, organized according to HMI server. Selection of an HMI application node can cause properties for the selected application to be rendered in the Properties panel 936, or can render the HMI application in a canvas 940.

[0105] FIG. 25 is a view of the Explorer panel 910 with the Devices view currently selected. The Device view is invoked by selecting the Devices icon 2502 in the explorer view control bar 908. The Devices view renders a Device navigation tree 2504 in the Explorer panel content area 1808. This tree 2504 comprises nodes representing devices (e.g., controllers, drives, motor control centers, etc.) that make up the control system project. Similar to the other Explorer views, information for a selected device can be rendered in the Properties panel 936 or on a canvas 940 by appropriate interaction with the device's node. FIG. 26 is a view of a canvas 940 on which information for an example controller is rendered in response to selection of a controller node from the Device navigation tree 2504. As shown in this example, information that can be rendered for a selected device can include, but is not limited to, a name and model of the device, a network address of the device, an overview description of the device, a firmware version currently installed on the device, a type of electronic keying, a connection type, or other such device information.

[0106] FIG. 27 is a view of the Explorer panel 910 with the Library view currently selected. The Library view is invoked by selecting the Library icon 2702 in the explorer view control bar 908. The Library view renders a Library navigation tree 2704 in the Explorer panel content area 1808. Library navigation tree 2704 comprises nodes representing software objects such as automation objects, add-on instructions, user-defined data types, device configurations, or other such objects. The Library view can include two or more tabs 1804 that allow the user to select sources of software objects to be viewed. In the illustrated example, tab 1804a renders objects associated with the current automation system project, tab 1804b renders objects available in a vendor library, and tab 1804c renders objects from an external source. Similar to the other Explorer views, information regarding a selected object can be rendered in the Properties panel 936 or on a canvas 940 by appropriate interaction with the object's node.

[0107] FIG. 28 is a view of the Explorer panel 910 with the Extensions view currently selected. The Extensions view is invoked by selecting the Extensions icon 2802 in the explorer view control bar 908. The Extensions view renders a list of software extensions currently installed on the IDE system 202, which may include, but are not limited to, dashboards, system viewers and designers, ladder logic editors, function block diagram editors, structured text editors, HMI screen editors, or other such extensions.

[0108] Some embodiments of IDE system's user interface component 204 can also support multi-instance states of the project development environment, such that the development environment can be distributed across multiple display devices. Such embodiments can support multi-instance workflows that help to orient the user within the development environment and that allow the user to easily locate relevant editors within the expanded and

distributed workspace, and to work fluidly across the multiple instances of the development interface 902.

[0109] FIGs. 29a and 29b depict an example distributed, multi-instance implementation of development interface 902. In this example, the development environment for an automation project currently being developed has been distributed across two monitors or other display devices, effectively expanding the development interface 902 across two separate but linked instances - development interface 902a (FIG. 29a) rendered on a left-side monitor and development interface 902b (FIG. 29b) rendered on a right-side monitor. In the illustrated example, the left-side interface 902a renders a first canvas 940a (and associated tab 932a) on which is displayed a control routine currently being developed. Interface 902a also renders the Explorer panel 910 and its associated explorer view control bar 908 in the left global panel area 922, a first instance of the Properties panel 936a in the right global panel area 928, and a first instance of an overlaid Layers panel 2902a adjacent to the Properties panel 936a. A first instance of the panel control bar 920a is anchored on the right edge of the interface 902a.

[0110] The right-side interface 902b renders two horizontally stacked canvases 940b and 940c (and their associated tabs 932a and 932b) containing two other aspects of the system project - a tag database and a parameter view, respectively. Second instances of the Properties panel 936b and Layers panel 2902b are rendered on the right-side of the interface 902b, and a second instance of the panel control bar 920b is anchored on the right edge of the interface 902b. In this example scenario, the user has opted to omit the Explorer panel 910 from the right global panel area of the second interface 902b.

[0111] Although only two instances of interface 902 are depicted in the example illustrated in FIGs. 29a-29b, the user interface component 204 can support expansion of the development interface 902 across any number of instances (e.g., if more than two display devices are available). Moreover, although the illustrated example depicts three opened canvases 940a-940c distributed across the two instances, any number of tabbed canvases 940 can be rendered on each instance of the interface 902.

[0112] The two interfaces 902a and 902b are extensions of one another, such that moving the cursor beyond the right boundary of left-side interface 902a causes the cursor to enter the right-side interface 902b via the left boundary of the right-side interface 902b, and vice versa. Thus, the user can fluidly traverse across the three canvases 940a-940c. In general, the user can configure panel visibility and layouts independently for each extended interface 902a and 902b. For example, the user may opt to render copies of the same global panel on both interface instances, or may choose to render a given panel visible on one interface while omitting the panel from the other interface.

[0113] To assist the user to easily navigate between the interface instances, particularly in scenarios in which

several tabbed canvases 940 are open, some embodiments of interface 902 can render an Available Tabs menu in response to selection of a suitable control (e.g., a control in the menu bar 904), which lists the tabs 932 that are currently open and available for selective focus. FIG. 30 is an example Available Tabs menu 3002 that can be invoked in such embodiments. Example menu 3002 lists the currently active canvases 940 according to name (e.g., Ladder 1, Tags, Parameters, etc.) and segregates the list according to the instance of interface 902 on which the respective canvases 940 currently reside. The list can be segregated vertically such that a first section 3004 lists the tabs 932 visible on the first instance of interface 902 and a second section 3006 lists the tabs 932 visible on the second instance. Selecting any of the tabs on the menu 3002 will cause the interface 902 to move the focus to the selected tab 936 (that is, bring the selected tab to the front of the workspace). By listing all active tabs in one menu 3002, a user can easily select a desired tab that may be located on an interface instance other than the one currently being viewed by the user, or that may be hidden under other overlaid canvases 940 or panels. This can mitigate the need to search through the distributed instances of interface 902 to locate a desired canvas 940.

[0114] Menu 3002 can also include other controls for manipulating the tabs 932. For example, a Consolidate menu option 3008 can cause all tab instances across the multiple interface instances to be moved to the interface instance currently being viewed (that is, the instance from which the Consolidate command was triggered). In some embodiments, performing this Consolidate function will also cause all extended instances of interface 902 to be closed, leaving only the currently viewed instance active.

[0115] A tab 932 and its associated canvas 940 can be moved from one instance of interface 902 to another by selecting and dragging the tab from its current instance of interface 902 to the target instance (e.g., a target instance on another display device). If a tab 932 is moved to an instance of interface 902 that already contains one or more visible canvases 940, the existing canvases will be resized to accommodate the addition of the canvas 940 associated with the relocated tab 932. In such cases, the canvases 940 can automatically determine a suitable configuration of horizontal and/or vertical stacking of the canvases 940 based on the current orientations of the preexisting tabs and the drop location of the relocated tab.

[0116] In some embodiments, layout and functionality of the development interface 902 can also be responsive to the size of the screen or display device on which the interface is rendered. The dimensions of the boundaries within which the interface 902 operates can be a function of the dimensions of the device's display screen, or may be set by the user by resizing the IDE system's development environment window. In either case, user interface component 204 can be configured to enable or disable certain functions of the development interface 902 based

on the size or aspect ratio of the interface's boundaries, and to reorganize elements of the development interface 902 as needed to fill the available horizontal and vertical viewport space as a function of available space.

[0117] In an example embodiment, development interface 902 can support multiple layout modes corresponding to respective ranges of screen or window widths. FIGs. 31a-31d are example instances of development interface 902 that accord to respective different layout modes as a function of available screen width.

[0118] FIG. 31a depicts a first layout mode suitable for scenarios in which there are no width restrictions. This first layout mode offers full support for all primary interface elements, as described above.

[0119] FIG. 31b depicts a second layout mode that may be initiated by user interface component 204 when the available screen width is below a first threshold width. According to this second layout mode, global panel sections (e.g., Properties panel 936) are removed, and pinned panels are prohibited (that is, all panels are rendered as overlay panels). Left and bottom panel support is disabled, and only global right overlay panels are permitted to be rendered. Only one panel is permitted to be rendered at a given time. Content panel visibility icons, which are normally rendered on the canvas's tool bar, are moved to the global panel control bar 920 (e.g., Layers visibility icon 3102). Support for multiple stacked canvases is disabled. The Explorer panel 910, including its associated explorer view control bar 908, is moved from the left side to the right side of the interface 902 adjacent to the global panel control bar 920.

[0120] FIG. 31c depicts a third layout mode that may be initiated by user interface component 204 when the available screen width is below a second threshold width that is smaller than the first threshold width. This third layout mode maintains all limitations and restrictions of the second layout mode. In addition, header elements are collapsed to reduce the number of selections visible at the same time. This includes collapsing the visible selection on menu bar 904 into a single selectable menu icon 3104, which can be selected to render the menu bar options as a drop-down list. Similarly, the selections on the tool bar 908 are collapsed into a single Tools icon 3108, which can be selected to render the tool bar selections in another drop-down list. Search bar 934 is also reduced to a selectable Search icon 3110. As a result of these consolidations, the total number of visible selections is reduced, thereby decluttering the limited development space.

[0121] The industrial IDE development interface 902 described herein offers a highly adaptable workspace layout that intelligently filters information and editing tools available to the user at a given time as a function of the user's current development task or focus, which allows desired information and editing tools relevant to the current development context to be located easily. In addition, the interface 902 affords the user a great deal of control over customization of the workspace layout, while main-

taining a clean and uncluttered development space that can be navigated easily. The IDE system 902 and its associated development interfaces 902 are suitable for developing multiple aspects of an industrial automation system - e.g., control programming, device configuration, alarm configuration, visualization screen development - within the same multi-content workspace, and can be used to develop projects ranging in scale from single controller systems to systems encompassing scores of controllers across different industrial facilities.

[0122] FIGs. 32a-35b illustrate various methodologies in accordance with one or more embodiments of the subject application. While, for purposes of simplicity of explanation, the one or more methodologies shown herein are shown and described as a series of acts, it is to be understood and appreciated that the subject innovation is not limited by the order of acts, as some acts may, in accordance therewith, occur in a different order and/or concurrently with other acts from that shown and described herein. For example, those skilled in the art will understand and appreciate that a methodology could alternatively be represented as a series of interrelated states or events, such as in a state diagram. Moreover, not all illustrated acts may be required to implement a methodology in accordance with the innovation. Furthermore, interaction diagram(s) may represent methodologies, or methods, in accordance with the subject disclosure when disparate entities enact disparate portions of the methodologies. Further yet, two or more of the disclosed example methods can be implemented in combination with each other, to accomplish one or more features or advantages described herein.

[0123] FIG. 32a illustrates a first part of an example methodology 3200a for customizing panel visibility and layout on a development interface of an industrial IDE system. Initially, at 3202, an industrial IDE interface is rendered comprising a workspace canvas and a global panel control bar pinned to an edge of the IDE development interface. The global panel control bar can comprise a set of visibility icons that control visibility of respective global panels supported by the industrial IDE system. In some embodiments, the development interface can comprise segregated global panel display areas - e.g., a left, right, and bottom global panel area - and the visibility icons can be organized on the global panel control bar according to the panel display area to which the respective panels have been designated.

[0124] At 3204, a determination is made as to whether a panel visibility icon has been selected from a left panel area of the global panel control bar. The left panel area is a section of the global panel control bar on which is rendered visibility icons corresponding to a subset of the global panels that have been designated to the left global panel area of the development interface. If a visibility icon has been selected from the left panel area of the global panel control bar (YES at step 3204), the methodology proceeds to step 3206, where a determination is made as to whether the panel corresponding to the visibility

icon selected at step 3204 has been set to be a pinned panel. For example, the panel may have been previously set to be pinned by a user via an appropriate interaction with a properties menu associated with the panel. If the panel has been set to be pinned (YES at step 3206), methodology proceeds to step 3208, where the panel corresponding to the visibility icon is rendered in the left global panel area of the development interface as a pinned panel. Alternatively, if the panel has not been set to be pinned (NO at step 3206), the methodology proceeds to step 3210, where the panel is rendered in the left global panel area as an overlay panel.

[0125] Once the panel has been rendered, or if no panel visibility icon has been selected from the left panel area of the global panel control bar (NO at step 3204), the methodology proceeds to the second part 3200b illustrated in FIG. 32b. At 3212, a determination is made as to whether a panel visibility icon has been selected from a bottom panel area of the global panel control bar. The bottom panel area is a section of the global panel control bar on which is rendered visibility icons corresponding to a subset of the global panels that have been designated to the bottom global panel area of the development interface. If a visibility icon has been selected from the bottom panel area (YES at step 3212), the methodology proceeds to step 3214, where a determination is made as to whether a panel corresponding to the visibility icon selected at step 3212 has been set to be a pinned panel. If the panel has been set to be pinned (YES at step 3214), the methodology proceeds to step 3216, where the panel corresponding to the selected visibility icon is rendered in the bottom global panel area of the development interface as a pinned panel. Alternatively, if the panel has not been set to be pinned (NO at step 3214), the methodology proceeds to step 3218, where the panel is rendered in the bottom global panel area as an overlay panel.

[0126] Once the panel has been rendered, or if no panel visibility icon has been selected from the bottom panel area of the global panel control bar (NO at step 3212), the methodology proceeds to the third part 3200c illustrated in FIG. 32c. At 3220, a determination is made as to whether a panel visibility icon has been selected from a right panel area of the global panel control bar. The right panel area is a section of the global panel control bar on which is rendered visibility icons corresponding to a subset of the global panels that have been designated to the right global panel area of the development interface. If a visibility icon has been selected from the right panel area (YES at step 3220), the methodology proceeds to step 3222, where a determination is made as to whether a panel corresponding to the visibility icon selected at step 3220 has been set to be a pinned panel. If the panel has been set to be pinned (YES at step 3222), the methodology proceeds to step 3224, where the panel corresponding to the selected visibility icon is rendered in the right global panel area of the development interface as a pinned panel. Alternatively, if the panel has not been set to be pinned (NO at step 3222), the methodology

proceeds to step 3226, where the panel is rendered in the right global panel area as an overlay panel.

[0127] Once the panel has been rendered, or if no panel visibility icon has been selected from the right panel area of the global panel control bar (NO at step 3222), the methodology returns to step 3202 and the methodology repeats.

[0128] FIG. 33a illustrates a first part of an example methodology 3300a for browsing and rendering aspects of an industrial automation project via interaction with an industrial IDE development interface. Initially, at 3302, an explorer panel is rendered on the development interface, where the explorer panel is configured to facilitate browsing and selecting of aspects of an industrial automation project (e.g., control programming or routines, HMI development screens, controller tag databases, industrial device parameter configurations, alarm configurations, etc.) to be rendered on the development interface. The explorer panel can comprise a set of selectable icons representing respective viewing categories supported by the explorer panel, where each viewing category defines content and formatting of selections to be presented in the explorer panel. In some embodiments, the explorer panel can be selectively rendered or hidden using the methodology described above in connection with FIGs. 32a-32c.

[0129] At 3304, selection of an icon representing one of the viewing categories from the set of supported viewing categories is received. Example viewing categories that can be selected in this manner can include, but are not limited to, a System view that lists components of the automation system project (e.g., control routines, tags, visualization applications or screens, alarms, etc.), an Application view that lists applications that make up the automation system project (e.g., control programming applications, HMI applications, etc.), a Devices view that lists devices that make up the automation system project, a Library view that lists software objects that make up the automation system project (e.g., automation objects, add-on instructions, user-defined data types, device configurations, etc.), and an Extensions view that lists software add-ons or extensions that have been installed on the industrial IDE system. Some or all of the content associated with these views can be rendered in a hierarchical format to allow users to more quickly and easily browse and locate a desired selection.

[0130] At 3306, in response to selection of the icon at step 3304, two or more tabs are rendered on the explorer panel, the two or more tabs representing respective two or more presentation formats for content within the viewing category corresponding to the selected icon. For example, selection of an Application view icon may cause the explorer panel to render two or more tabs representing respective different types of applications that can be explored (e.g., controller applications, HMI applications, etc.). In another example, selection of a Library view can cause the explorer panel to render two or more tabs representing respective sources of software objects that can

be explored.

[0131] At 3308, selectable icons are rendered on a content window of the explorer panel, where the icons correspond to the viewing category and a first presentation format corresponding to a first tab of the two or more tabs rendered at step 3306. The selectable icons - which may be graphical, text-based, or a combination of both - represent aspects of the automation system project that can be browsed and selected for presentation in the development interfaces may workspace or canvas.

[0132] The methodology continues with the second part 3300b illustrated in FIG. 33b. At 3310, a determination is made as to whether a second tab of the two or more tabs rendered at step 3306 has been selected. If the second tab has been selected (YES at step 3310), the methodology proceeds to step 3312, where selectable icons - which may include some or all of the selectable icons represented at step 3308 or a different set of icons - are rendered in the content window of the explorer panel in a second presentation format corresponding to the second tab.

[0133] If the second tab is not selected (NO at step 3310) or after the icons have been rendered in the second format at step 3312, the methodology proceeds to step 3314, where a determination is made as to whether an icon is selected from the content window of the explorer panel. If an icon has been selected (YES at step 3314), the methodology proceeds to step 3316, where an aspect of the automation system project corresponding to the icon is rendered. The aspect may be, for example, a ladder logic routine, a structure text program, a function block diagram, an HMI development screen, an alarm configuration screen, a device parameter configuration screen, an engineering drawing or schematic, or another such aspect.

[0134] FIG. 34a illustrates a first part of an example methodology 3400a for manipulating workspace canvases within an industrial IDE development interface. Initially, at 3402, two different aspects of an automation system project are rendered in respective two tabbed canvases of an industrial IDE development interface. The two tabbed canvases are initially rendered such that a first of the two canvases is overlaid over a second of the two canvases such that content of only one canvas is visible at a given time, and the visible content can be selected by selecting the appropriate tab. Project aspects that can be rendered in these tabbed canvases can include, but are not limited to, control programming, tag databases, device configurations, HMI development screens, alarm configurations, or other such content.

[0135] At 3404, a determination is made as to whether a command to stack the canvases horizontally has been received. If such a command is received (YES at step 3404), the methodology proceeds to step 3406, where the two canvases are rendered such that content of the two canvases is displayed simultaneously and the canvases are arranged horizontally. Alternatively, if the command to stack the canvases horizontally is not received

(NO at step 3404), the methodology proceeds to step 3408, where a determination is made as to whether a command to stack the canvases vertically has been received. If such a command is received (YES at step 3408) the methodology proceeds to step 3410, where the two canvases are rendered that content of the two canvases is displayed simultaneously and the canvases are arranged horizontally.

[0136] The methodology then continues with the second part 3400b illustrated in FIG. 34b. At 3412, a determination is made as to whether a command to distribute the tabbed canvases across two display devices has been received. This command may be received in implementations in which the interface display is extended across two display devices to expand the usable workspace. If the command to distribute the tabbed canvases is received (YES at step 3412), the methodology proceeds step 3414, where the first canvas is rendered on a first instance of the development interface on a first display device and the second canvas is rendered on a second instance of the development interface on a second display device. While the canvases are distributed in this manner, a determination is made at step 3416 as to whether a command to consolidate the tabbed canvases is received. If such a command is received (YES at step 3416), the methodology proceeds to step 3418, where the two canvases are consolidated onto one of the two instances of the development interface from which the command to consolidate was received. The methodology then returns to step 3402.

[0137] If the command to distribute the tabbed canvases is not received at step 3412 (NO at step 3412) - that is, the canvases are still consolidated on a single instance of the interface display and are stacked horizontally or vertically - the methodology proceeds to the third part 3400c illustrated in FIG. 34c. At 3420, a determination is made as to whether a command to overlay the tabbed canvases is received. If no such command is received (NO at step 3420), the methodology returns to step 3404. Alternatively, if the command to overlay the canvases is received (YES at step 3420), the methodology returns to step 3402, where the canvases are again rendered as overlays.

[0138] In some embodiments, the canvas manipulation methodology of FIGs. 34a-34c can be combined with one or both of the methodologies described above in connection with FIGs. 32a-32c and 33a-33b.

[0139] FIG. 35a illustrates a first part of an example methodology 3500a for automatically curating a set of available project editing tools by an industrial IDE development interface based on a current development task being performed by a user. Initially, at 3502, a global panel control bar is rendered on an industrial IDE development interface comprising one or more workspace canvases. The global panel control bar can be pinned to an edge of the development interface, and can comprise a first set of visibility icons that correspond to a first set of global panels supported by the industrial IDE that are

applicable to all design contexts of the industrial IDE.

[0140] At 3504, a current automation project development task being performed via the one or more workspace canvases is determined. The task can be determined, for example, based on content of the workspace canvas that currently has focus within the development interface. The task may be, for example, ladder logic control programming, structured text control programming, function block diagram control programming, HMI screen development, device configuration, controller tag editing, alarm configuration, or other such tasks.

[0141] At 3506, a second set of visibility icons is rendered on the development interface. The second set of visibility icons correspond to one or more content panels supported by the industrial IDE that are not globally applicable but are applicable to the current development task determined at step 3504.

[0142] The methodology continues with the second part 3500b illustrated in FIG. 35b. At 3508, selection of a visibility icon from among the first or second set of visibility icons is received. At 3510, a determination as to whether a panel corresponding to the icon selected at step 3508 is set to be a pinned panel. If the selected panel is set to be pinned (YES at step 3510), the methodology proceeds to step 3512, where the panel corresponding to the selected icon is rendered on the development interface as a pinned panel. Alternatively, if the selected panel is not set to be pinned (NO at step 3510), the methodology proceeds to step 3514, where the panel corresponding to the selected icon is rendered on the development interface as an overlay panel.

[0143] In some embodiments, the methodology described in connection with FIGs. 35a-35b can be combined with one or more of the other methodologies described herein.

[0144] Embodiments, systems, and components described herein, as well as control systems and automation environments in which various aspects set forth in the subject specification can be carried out, can include computer or network components such as servers, clients, programmable logic controllers (PLCs), automation controllers, communications modules, mobile computers, on-board computers for mobile vehicles, wireless components, control components and so forth which are capable of interacting across a network. Computers and servers include one or more processors-electronic integrated circuits that perform logic operations employing electric signals-configured to execute instructions stored in media such as random access memory (RAM), read only memory (ROM), a hard drives, as well as removable memory devices, which can include memory sticks, memory cards, flash drives, external hard drives, and so on.

[0145] Similarly, the term PLC or automation controller as used herein can include functionality that can be shared across multiple components, systems, and/or networks. As an example, one or more PLCs or automation controllers can communicate and cooperate with var-

ious network devices across the network. This can include substantially any type of control, communications module, computer, Input/Output (I/O) device, sensor, actuator, and human machine interface (HMI) that communicate *via* the network, which includes control, automation, and/or public networks. The PLC or automation controller can also communicate to and control various other devices such as standard or safety-rated I/O modules including analog, digital, programmed/intelligent I/O modules, other programmable controllers, communications modules, sensors, actuators, output devices, and the like.

[0146] The network can include public networks such as the internet, intranets, and automation networks such as control and information protocol (CIP) networks including DeviceNet, ControlNet, safety networks, and Ethernet/IP. Other networks include Ethernet, DH/DH+, Remote I/O, Fieldbus, Modbus, Profibus, CAN, wireless networks, serial protocols, and so forth. In addition, the network devices can include various possibilities (hardware and/or software components). These include components such as switches with virtual local area network (VLAN) capability, LANs, WANs, proxies, gateways, routers, firewalls, virtual private network (VPN) devices, servers, clients, computers, configuration tools, monitoring tools, and/or other devices.

[0147] In order to provide a context for the various aspects of the disclosed subject matter, FIGs. 36 and 37 as well as the following discussion are intended to provide a brief, general description of a suitable environment in which the various aspects of the disclosed subject matter may be implemented. While the embodiments have been described above in the general context of computer-executable instructions that can run on one or more computers, those skilled in the art will recognize that the embodiments can be also implemented in combination with other program modules and/or as a combination of hardware and software.

[0148] Generally, program modules include routines, programs, components, data structures, etc., that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the inventive methods can be practiced with other computer system configurations, including single-processor or multiprocessor computer systems, minicomputers, mainframe computers, Internet of Things (IoT) devices, distributed computing systems, as well as personal computers, hand-held computing devices, microprocessor-based or programmable consumer electronics, and the like, each of which can be operatively coupled to one or more associated devices.

[0149] The illustrated embodiments herein can be also practiced in distributed computing environments where certain tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules can be located in both local and remote memory storage devices.

[0150] Computing devices typically include a variety of media, which can include computer-readable storage media, machine-readable storage media, and/or communications media, which two terms are used herein differently from one another as follows. Computer-readable storage media or machine-readable storage media can be any available storage media that can be accessed by the computer and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer-readable storage media or machine-readable storage media can be implemented in connection with any method or technology for storage of information such as computer-readable or machine-readable instructions, program modules, structured data or unstructured data.

[0151] Computer-readable storage media can include, but are not limited to, random access memory (RAM), read only memory (ROM), electrically erasable programmable read only memory (EEPROM), flash memory or other memory technology, compact disk read only memory (CD-ROM), digital versatile disk (DVD), Blu-ray disc (BD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, solid state drives or other solid state storage devices, or other tangible and/or non-transitory media which can be used to store desired information. In this regard, the terms "tangible" or "non-transitory" herein as applied to storage, memory or computer-readable media, are to be understood to exclude only propagating transitory signals per se as modifiers and do not relinquish rights to all standard storage, memory or computer-readable media that are not only propagating transitory signals per se.

[0152] Computer-readable storage media can be accessed by one or more local or remote computing devices, e.g., via access requests, queries or other data retrieval protocols, for a variety of operations with respect to the information stored by the medium.

[0153] Communications media typically embody computer-readable instructions, data structures, program modules or other structured or unstructured data in a data signal such as a modulated data signal, e.g., a carrier wave or other transport mechanism, and includes any information delivery or transport media. The term "modulated data signal" or signals refers to a signal that has one or more of its characteristics set or changed in such a manner as to encode information in one or more signals. By way of example, and not limitation, communication media include wired media, such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media.

[0154] With reference again to FIG. 36, the example environment 3600 for implementing various embodiments of the aspects described herein includes a computer 3602, the computer 3602 including a processing unit 3604, a system memory 3606 and a system bus 3608. The system bus 3608 couples system components including, but not limited to, the system memory 3606 to

the processing unit 3604. The processing unit 3604 can be any of various commercially available processors. Dual microprocessors and other multi-processor architectures can also be employed as the processing unit 3604.

[0155] The system bus 3608 can be any of several types of bus structure that can further interconnect to a memory bus (with or without a memory controller), a peripheral bus, and a local bus using any of a variety of commercially available bus architectures. The system memory 3606 includes ROM 3610 and RAM 3612. A basic input/output system (BIOS) can be stored in a non-volatile memory such as ROM, erasable programmable read only memory (EPROM), EEPROM, which BIOS contains the basic routines that help to transfer information between elements within the computer 3602, such as during startup. The RAM 3612 can also include a high-speed RAM such as static RAM for caching data.

[0156] The computer 3602 further includes an internal hard disk drive (HDD) 3614 (e.g., EIDE, SATA), one or more external storage devices 3616 (e.g., a magnetic floppy disk drive (FDD) 3616, a memory stick or flash drive reader, a memory card reader, etc.) and an optical disk drive 3620 (e.g., which can read or write from a CD-ROM disc, a DVD, a BD, etc.). While the internal HDD 3614 is illustrated as located within the computer 3602, the internal HDD 3614 can also be configured for external use in a suitable chassis (not shown). Additionally, while not shown in environment 3600, a solid state drive (SSD) could be used in addition to, or in place of, an HDD 3614. The HDD 3614, external storage device(s) 3616 and optical disk drive 3620 can be connected to the system bus 3608 by an HDD interface 3624, an external storage interface 3626 and an optical drive interface 3628, respectively. The interface 3624 for external drive implementations can include at least one or both of Universal Serial Bus (USB) and Institute of Electrical and Electronics Engineers (IEEE) 1394 interface technologies. Other external drive connection technologies are within contemplation of the embodiments described herein.

[0157] The drives and their associated computer-readable storage media provide nonvolatile storage of data, data structures, computer-executable instructions, and so forth. For the computer 3602, the drives and storage media accommodate the storage of any data in a suitable digital format. Although the description of computer-readable storage media above refers to respective types of storage devices, it should be appreciated by those skilled in the art that other types of storage media which are readable by a computer, whether presently existing or developed in the future, could also be used in the example operating environment, and further, that any such storage media can contain computer-executable instructions for performing the methods described herein.

[0158] A number of program modules can be stored in the drives and RAM 3612, including an operating system 3630, one or more application programs 3632, other program modules 3634 and program data 3636. All or portions of the operating system, applications, modules,

and/or data can also be cached in the RAM 3612. The systems and methods described herein can be implemented utilizing various commercially available operating systems or combinations of operating systems.

[0159] Computer 3602 can optionally comprise emulation technologies. For example, a hypervisor (not shown) or other intermediary can emulate a hardware environment for operating system 3630, and the emulated hardware can optionally be different from the hardware illustrated in FIG. 36. In such an embodiment, operating system 3630 can comprise one virtual machine (VM) of multiple VMs hosted at computer 3602. Furthermore, operating system 3630 can provide runtime environments, such as the Java runtime environment or the .NET framework, for application programs 3632. Runtime environments are consistent execution environments that allow application programs 3632 to run on any operating system that includes the runtime environment. Similarly, operating system 3630 can support containers, and application programs 3632 can be in the form of containers, which are lightweight, standalone, executable packages of software that include, e.g., code, runtime, system tools, system libraries and settings for an application.

[0160] Further, computer 3602 can be enable with a security module, such as a trusted processing module (TPM). For instance with a TPM, boot components hash next in time boot components, and wait for a match of results to secured values, before loading a next boot component. This process can take place at any layer in the code execution stack of computer 3602, e.g., applied at the application execution level or at the operating system (OS) kernel level, thereby enabling security at any level of code execution.

[0161] A user can enter commands and information into the computer 3602 through one or more wired/wireless input devices, e.g., a keyboard 3638, a touch screen 3640, and a pointing device, such as a mouse 3642. Other input devices (not shown) can include a microphone, an infrared (IR) remote control, a radio frequency (RF) remote control, or other remote control, a joystick, a virtual reality controller and/or virtual reality headset, a game pad, a stylus pen, an image input device, e.g., camera(s), a gesture sensor input device, a vision movement sensor input device, an emotion or facial detection device, a biometric input device, e.g., fingerprint or iris scanner, or the like. These and other input devices are often connected to the processing unit 3604 through an input device interface 3644 that can be coupled to the system bus 3608, but can be connected by other interfaces, such as a parallel port, an IEEE 1394 serial port, a game port, a USB port, an IR interface, a BLUETOOTH® interface, etc.

[0162] A monitor 3644 or other type of display device can be also connected to the system bus 3608 via an interface, such as a video adapter 3648. In addition to the monitor 3644, a computer typically includes other peripheral output devices (not shown), such as speakers,

printers, etc.

[0163] The computer 3602 can operate in a networked environment using logical connections via wired and/or wireless communications to one or more remote computers, such as a remote computer(s) 3648. The remote computer(s) 3648 can be a workstation, a server computer, a router, a personal computer, portable computer, microprocessor-based entertainment appliance, a peer device or other common network node, and typically includes many or all of the elements described relative to the computer 3602, although, for purposes of brevity, only a memory/storage device 3650 is illustrated. The logical connections depicted include wired/wireless connectivity to a local area network (LAN) 3652 and/or larger networks, e.g., a wide area network (WAN) 3654. Such LAN and WAN networking environments are commonplace in offices and companies, and facilitate enterprise-wide computer networks, such as intranets, all of which can connect to a global communications network, e.g., the Internet.

[0164] When used in a LAN networking environment, the computer 3602 can be connected to the local network 3652 through a wired and/or wireless communication network interface or adapter 3656. The adapter 3656 can facilitate wired or wireless communication to the LAN 3652, which can also include a wireless access point (AP) disposed thereon for communicating with the adapter 3656 in a wireless mode.

[0165] When used in a WAN networking environment, the computer 3602 can include a modem 3658 or can be connected to a communications server on the WAN 3654 via other means for establishing communications over the WAN 3654, such as by way of the Internet. The modem 3658, which can be internal or external and a wired or wireless device, can be connected to the system bus 3608 via the input device interface 3642. In a networked environment, program modules depicted relative to the computer 3602 or portions thereof, can be stored in the remote memory/storage device 3650. It will be appreciated that the network connections shown are example and other means of establishing a communications link between the computers can be used.

[0166] When used in either a LAN or WAN networking environment, the computer 3602 can access cloud storage systems or other network-based storage systems in addition to, or in place of, external storage devices 3616 as described above. Generally, a connection between the computer 3602 and a cloud storage system can be established over a LAN 3652 or WAN 3654 e.g., by the adapter 3656 or modem 3658, respectively. Upon connecting the computer 3602 to an associated cloud storage system, the external storage interface 3626 can, with the aid of the adapter 3656 and/or modem 3658, manage storage provided by the cloud storage system as it would other types of external storage. For instance, the external storage interface 3626 can be configured to provide access to cloud storage sources as if those sources were physically connected to the computer 3602.

[0167] The computer 3602 can be operable to communicate with any wireless devices or entities operatively disposed in wireless communication, e.g., a printer, scanner, desktop and/or portable computer, portable data assistant, communications satellite, any piece of equipment or location associated with a wirelessly detectable tag (e.g., a kiosk, news stand, store shelf, etc.), and telephone. This can include Wireless Fidelity (Wi-Fi) and BLUETOOTH® wireless technologies. Thus, the communication can be a predefined structure as with a conventional network or simply an ad hoc communication between at least two devices.

[0168] FIG. 37 is a schematic block diagram of a sample computing environment 1800 with which the disclosed subject matter can interact. The sample computing environment 3700 includes one or more client(s) 3702. The client(s) 3702 can be hardware and/or software (e.g., threads, processes, computing devices). The sample computing environment 3700 also includes one or more server(s) 3704. The server(s) 3704 can also be hardware and/or software (e.g., threads, processes, computing devices). The servers 3704 can house threads to perform transformations by employing one or more embodiments as described herein, for example. One possible communication between a client 3702 and servers 3704 can be in the form of a data packet adapted to be transmitted between two or more computer processes. The sample computing environment 3700 includes a communication framework 3706 that can be employed to facilitate communications between the client(s) 3702 and the server(s) 3704. The client(s) 3702 are operably connected to one or more client data store(s) 3708 that can be employed to store information local to the client(s) 3702. Similarly, the server(s) 3704 are operably connected to one or more server data store(s) 3710 that can be employed to store information local to the servers 3704.

[0169] What has been described above includes examples of the subject innovation. It is, of course, not possible to describe every conceivable combination of components or methodologies for purposes of describing the disclosed subject matter, but one of ordinary skill in the art may recognize that many further combinations and permutations of the subject innovation are possible. Accordingly, the disclosed subject matter is intended to embrace all such alterations, modifications, and variations that fall within the spirit and scope of the appended claims.

[0170] In particular and in regard to the various functions performed by the above described components, devices, circuits, systems and the like, the terms (including a reference to a "means") used to describe such components are intended to correspond, unless otherwise indicated, to any component which performs the specified function of the described component (e.g., a functional equivalent), even though not structurally equivalent to the disclosed structure, which performs the function in the herein illustrated exemplary aspects of the disclosed

subject matter. In this regard, it will also be recognized that the disclosed subject matter includes a system as well as a computer-readable medium having computer-executable instructions for performing the acts and/or events of the various methods of the disclosed subject matter. 5

[0171] In addition, while a particular feature of the disclosed subject matter may have been disclosed with respect to only one of several implementations, such feature may be combined with one or more other features of the other implementations as may be desired and advantageous for any given or particular application. Furthermore, to the extent that the terms "includes," and "including" and variants thereof are used in either the detailed description or the claims, these terms are intended to be inclusive in a manner similar to the term "comprising." 10

[0172] In this application, the word "exemplary" is used to mean serving as an example, instance, or illustration. Any aspect or design described herein as "exemplary" is not necessarily to be construed as preferred or advantageous over other aspects or designs. Rather, use of the word exemplary is intended to present concepts in a concrete fashion. 20

[0173] Various aspects or features described herein may be implemented as a method, apparatus, or article of manufacture using standard programming and/or engineering techniques. The term "article of manufacture" as used herein is intended to encompass a computer program accessible from any computer-readable device, carrier, or media. For example, computer readable media can include but are not limited to magnetic storage devices (e.g., hard disk, floppy disk, magnetic strips...), optical disks [e.g., compact disk (CD), digital versatile disk (DVD)...], smart cards, and flash memory devices (e.g., card, stick, key drive...). 25 30 35

[0174] The following is a list of further preferred embodiments of the invention:

Embodiment 1. A system for developing industrial applications, comprising: 40

a memory that stores executable components; and

a processor, operatively coupled to the memory, that executes the executable components, the executable components comprising: 45

a user interface component configured to render an industrial integrated development environment (IDE) development interface and to receive, via interaction with the development interface, industrial design input that defines aspects of an industrial automation project, wherein the development interface comprises 50 55

one or more workspace canvases configured to facilitate development of a selected aspect of the industrial automation project, and

a global panel control bar comprising visibility icons corresponding to respective panels available to be invoked on the development interface; and

a project generation component configured to generate system project data based on the industrial design input,

wherein

the interface display comprises a left global panel area, a right global panel area, and a bottom global panel area,

the respective panels are designated to one of the left global panel area, the right global panel area, or the bottom global panel area, and

the user interface component is further configured to, in response to selection of a visibility icon, of the visibility icons, toggle a visibility of a panel corresponding to the visibility icon in one of the left global panel area, the right global panel area, or the bottom global panel area.

Embodiment 2. The system of embodiment 1, wherein the respective panels comprise controls that allow the panels to be individually configured as one of a pinned panel or an overlay panel, and the user interface component is further configured to:

in response to the selection of the visibility icon and a determination that the corresponding panel is a pinned panel, render the corresponding panel as being pinned to a background of the development interface, and

in response to the selection of the visibility icon and a determination that the corresponding panel is an overlay panel, render the corresponding global panel as an overlay.

Embodiment 3. The system of embodiment 2, wherein rendering a pinned panel in one of the left global panel area, the right global panel area, or the bottom global panel area causes the one or more workspace canvases to resize to accommodate placement of the pinned panel, and

rendering an overlay panel in one of the left global panel area, the right global panel area, or the bottom global panel area causes the overlay panel to be rendered over a portion of the one or more workspace canvases.

Embodiment 4. The system of embodiment 2, wherein rendering an overlay panel in one of the left global panel area, the right global panel area, or the bottom global panel area in which a pinned panel is visible causes the overlay panel to be rendered over at least a portion of the pinned panel.

Embodiment 5. The system of embodiment 1, wherein the visibility icons are segregated on the global panel control bar into groups corresponding to the left global panel area, the right global panel area, and the bottom global panel area, and the visibility icon is assigned to one of the groups according to which of the left global panel area, the right global panel area, or the bottom global panel area will render the panel.

Embodiment 6. The system of embodiment 2, wherein the user interface component is further configured to, in response to receipt of input data indicative of a dragging of a visibility icon, of the visibility icons, from the global panel control bar to one of the left global panel area, the right global panel area, or the bottom global panel area, render a panel corresponding to the visibility icon on the one of the left global panel area, the right global panel area, or the bottom global panel area as a pinned panel.

Embodiment 7. The system of embodiment 2, wherein the user interface component is further configured to render multiple pinned panels in a same global panel area in a vertically stacked arrangement, and the same global panel area is one of the left global panel area, the right global panel area, or the bottom global panel area.

Embodiment 8. The system of embodiment 7, wherein the user interface component is further configured to, in response to receipt of an instruction to collapse one of the multiple pinned panels, collapse the one of the multiple pinned panels vertically.

Embodiment 9. The system of embodiment 1, wherein the visibility icons are first visibility icons, and the user interface component is further configured to, in response to invoking a development task on a workspace canvas, of the one or more workspace canvases, render one or more second visibility icons on the workspace canvas that control visibility of re-

spective one or more content panels that are relevant to the development task.

Embodiment 10. The system of embodiment 1, wherein the one or more workspace canvases comprise multiple tabbed canvases, and the user interface component is further configured to arrange the multiple canvases, in accordance with user input, to be one or more of vertically stacked, horizontally stacked, or overlaid.

Embodiment 11. The system of embodiment 10, wherein the user interface is further configured to extend the development interface across multiple display devices to yield multiple instance of the development interface, and to distribute the multiple tabbed interfaces across the multiple instances.

Embodiment 12. The system of embodiment 1, wherein the panel is at least one of an explorer panel that facilitates browsing of aspects of the industrial automation project, a properties panel that renders property information for a selected element within the one or more workspace canvases, an online panel that renders communication statistics for the system, a cross reference panel that renders cross reference information for a selected element within the one or more workspace canvases, an output panel that renders output statistics, an errors panel that renders development or runtime errors, or a toolbox panel that renders selectable editing tools.

Embodiment 13. A method for curating an industrial development workspace, comprising:

displaying, by an industrial integrated development environment (IDE) system comprising a processor, a development interface on a client device, wherein the displaying comprises:

displaying one or more workspace canvases on which respective development tasks are performed, and

displaying a global panel control bar comprising visibility icons corresponding to respective panels available to be invoked on the development interface;

receiving, by the industrial IDE system, selection of a visibility icon, of the visibility icons, via interaction with the development interface; and

in response to the receiving, toggling, by the industrial IDE system, a visibility of a panel corresponding to the visibility icon, wherein the toggling comprises adding the panel to or removing

the panel from one of a left global panel area, a right global panel area, or a bottom global panel area of the development interface.

Embodiment 14. The method of embodiment 13, wherein the toggling comprises:

in response to the selection of the visibility icon and a determination that the panel is set to be a pinned panel, rendering the panel as being pinned to a background of the development interface, and
in response to the selection of the visibility icon and a determination that the panel is set to be an overlay panel, rendering the panel as an overlay.

Embodiment 15. The method of embodiment 14, further comprising:

in response to determining that a pinned panel has been rendered visible in one of the left global panel area, the right global panel area, or the bottom global panel area, resizing, by the industrial IDE system, the one or more workspace canvases to accommodate placement of the pinned panel, and

in response to determining that an overlay panel has been rendered visible in one of the left global panel area, the right global panel area, or the bottom global panel area, overlaying the overlay panel over a portion of the one or more workspace canvases.

Embodiment 16. The method of embodiment 14, further comprising, in response to determining that multiple pinned panels have been rendered visible in a same global panel area, rendering the multiple pinned panels in a vertically stacked arrangement within the same global panel area.

Embodiment 17. The method of embodiment 16, further comprising, in response to receipt of an instruction to collapse one of the multiple pinned panels, collapsing the one of the multiple pinned panels vertically.

Embodiment 18. The method of embodiment 13, wherein the visibility icons are first visibility icons, and the method further comprises:
in response to invoking a development task on a workspace canvas, of the one or more workspace canvases, rendering one or more second visibility icons on the workspace canvas that control visibility of respective one or more content panels that are relevant to the development task.

Embodiment 19. A non-transitory computer-readable medium having stored thereon instructions that, in response to execution, cause an industrial integrated development environment (IDE) system comprising a processor to perform operations, the operations comprising:

displaying a development interface for the industrial IDE system on a client device, wherein the displaying comprises:

displaying one or more workspace canvases on which respective development tasks are performed, and

displaying a global panel control bar comprising visibility icons corresponding to respective panels available to be invoked on the development interface;

receiving selection of a visibility icon, of the visibility icons, via interaction with the development interface; and

in response to the receiving, toggling a visibility of a panel corresponding to the visibility icon, wherein the toggling comprises adding the panel to or removing the panel from one of a left global panel area, a right global panel area, or a bottom global panel area of the development interface.

Embodiment 20. The non-transitory computer-readable medium of embodiment 19, wherein the toggling comprises:

in response to the selection of the visibility icon and a determination that the panel is set to be a pinned panel, rendering the panel as being pinned to a background of the development interface, and

in response to the selection of the visibility icon and a determination that the panel is set to be an overlay panel, rendering the panel as an overlay.

Claims

1. A system for developing industrial applications, comprising:

a memory that stores executable components; and

a processor, operatively coupled to the memory, that executes the executable components, the executable components comprising:

- a user interface component configured to render an industrial integrated development environment (IDE) development interface and to receive, via interaction with the development interface, industrial design input that defines aspects of an industrial automation project, wherein the development interface comprises
- one or more workspace canvases configured to facilitate development of a selected aspect of the industrial automation project, and
 - a global panel control bar comprising visibility icons corresponding to respective panels available to be invoked on the development interface; and
- a project generation component configured to generate system project data based on the industrial design input,
- wherein
- the interface display comprises a left global panel area, a right global panel area, and a bottom global panel area,
- the respective panels are designated to one of the left global panel area, the right global panel area, or the bottom global panel area, and
- the user interface component is further configured to, in response to selection of a visibility icon, of the visibility icons, toggle a visibility of a panel corresponding to the visibility icon in one of the left global panel area, the right global panel area, or the bottom global panel area.
2. The system of claim 1, wherein the respective panels comprise controls that allow the panels to be individually configured as one of a pinned panel or an overlay panel, and the user interface component is further configured to:
 - in response to the selection of the visibility icon and a determination that the corresponding panel is a pinned panel, render the corresponding panel as being pinned to a background of the development interface, and
 - in response to the selection of the visibility icon and a determination that the corresponding panel is an overlay panel, render the corresponding global panel as an overlay.
 3. The system of claim 2, wherein rendering a pinned panel in one of the left global panel area, the right global panel area, or the bottom global panel area causes the one or more workspace canvases to resize to accommodate placement of the pinned panel, and rendering an overlay panel in one of the left global panel area, the right global panel area, or the bottom global panel area causes the overlay panel to be rendered over a portion of the one or more workspace canvases; and/or wherein rendering an overlay panel in one of the left global panel area, the right global panel area, or the bottom global panel area in which a pinned panel is visible causes the overlay panel to be rendered over at least a portion of the pinned panel.
 4. The system of one of claims 1 to 3, wherein the visibility icons are segregated on the global panel control bar into groups corresponding to the left global panel area, the right global panel area, and the bottom global panel area, and the visibility icon is assigned to one of the groups according to which of the left global panel area, the right global panel area, or the bottom global panel area will render the panel.
 5. The system of claim 2, wherein the user interface component is further configured to, in response to receipt of input data indicative of a dragging of a visibility icon, of the visibility icons, from the global panel control bar to one of the left global panel area, the right global panel area, or the bottom global panel area, render a panel corresponding to the visibility icon on the one of the left global panel area, the right global panel area, or the bottom global panel area as a pinned panel.
 6. The system of claim 2, wherein the user interface component is further configured to render multiple pinned panels in a same global panel area in a vertically stacked arrangement, and the same global panel area is one of the left global panel area, the right global panel area, or the bottom global panel area, wherein the user interface component is further configured to, in response to receipt of an instruction to collapse one of the multiple pinned panels, collapse the one of the multiple pinned panels vertically.
 7. The system of one of claims 1 to 6, wherein the visibility icons are first visibility icons, and the user interface component is further configured to, in response to invoking a development task on a workspace canvas, of the one or more workspace canvases, render one or more second visibility icons on the workspace canvas that control visibility of respective one or more content panels that are relevant to the development task.
 8. The system of one of claims 1 to 7, wherein the one or more workspace canvases comprise multiple tabbed canvases, and the user interface component is further configured

to arrange the multiple canvases, in accordance with user input, to be one or more of vertically stacked, horizontally stacked, or overlaid, wherein the user interface is further configured to extend the development interface across multiple display devices to yield multiple instance of the development interface, and to distribute the multiple tabbed interfaces across the multiple instances.

9. The system of one of claims 1 to 8, wherein the panel is at least one of an explorer panel that facilitates browsing of aspects of the industrial automation project, a properties panel that renders property information for a selected element within the one or more workspace canvases, an online panel that renders communication statistics for the system, a cross reference panel that renders cross reference information for a selected element within the one or more workspace canvases, an output panel that renders output statistics, an errors panel that renders development or runtime errors, or a toolbox panel that renders selectable editing tools.

10. A method for curating an industrial development workspace, comprising:

displaying, by an industrial integrated development environment (IDE) system comprising a processor, a development interface on a client device, wherein the displaying comprises:

displaying one or more workspace canvases on which respective development tasks are performed, and
displaying a global panel control bar comprising visibility icons corresponding to respective panels available to be invoked on the development interface;

receiving, by the industrial IDE system, selection of a visibility icon, of the visibility icons, via interaction with the development interface; and
in response to the receiving, toggling, by the industrial IDE system, a visibility of a panel corresponding to the visibility icon, wherein the toggling comprises adding the panel to or removing the panel from one of a left global panel area, a right global panel area, or a bottom global panel area of the development interface.

11. The method of claim 10, wherein the toggling comprises:

in response to the selection of the visibility icon and a determination that the panel is set to be a pinned panel, rendering the panel as being pinned to a background of the development interface, and

in response to the selection of the visibility icon and a determination that the panel is set to be an overlay panel, rendering the panel as an overlay.

12. The method of claim 11, further comprising:

in response to determining that a pinned panel has been rendered visible in one of the left global panel area, the right global panel area, or the bottom global panel area, resizing, by the industrial IDE system, the one or more workspace canvases to accommodate placement of the pinned panel, and in response to determining that an overlay panel has been rendered visible in one of the left global panel area, the right global panel area, or the bottom global panel area, overlaying the overlay panel over a portion of the one or more workspace canvases; and/or
in response to determining that multiple pinned panels have been rendered visible in a same global panel area, rendering the multiple pinned panels in a vertically stacked arrangement within the same global panel area, and in response to receipt of an instruction to collapse one of the multiple pinned panels, collapsing the one of the multiple pinned panels vertically.

13. The method of one of claims 10 to 12, wherein the visibility icons are first visibility icons, and the method further comprises:

in response to invoking a development task on a workspace canvas, of the one or more workspace canvases, rendering one or more second visibility icons on the workspace canvas that control visibility of respective one or more content panels that are relevant to the development task.

14. A non-transitory computer-readable medium having stored thereon instructions that, in response to execution, cause an industrial integrated development environment (IDE) system comprising a processor to perform operations, the operations comprising:

displaying a development interface for the industrial IDE system on a client device, wherein the displaying comprises:

displaying one or more workspace canvases on which respective development tasks are performed, and
displaying a global panel control bar comprising visibility icons corresponding to respective panels available to be invoked on the development interface;

receiving selection of a visibility icon, of the visibility icons, via interaction with the development

interface; and
in response to the receiving, toggling a visibility
of a panel corresponding to the visibility icon,
wherein the toggling comprises adding the panel
to or removing the panel from one of a left global
panel area, a right global panel area, or a bottom
global panel area of the development interface. 5

15. The non-transitory computer-readable medium of
claim 14, wherein the toggling comprises: 10

in response to the selection of the visibility icon
and a determination that the panel is set to be
a pinned panel, rendering the panel as being
pinned to a background of the development in-
terface, and 15
in response to the selection of the visibility icon
and a determination that the panel is set to be
an overlay panel, rendering the panel as an
overlay. 20

25

30

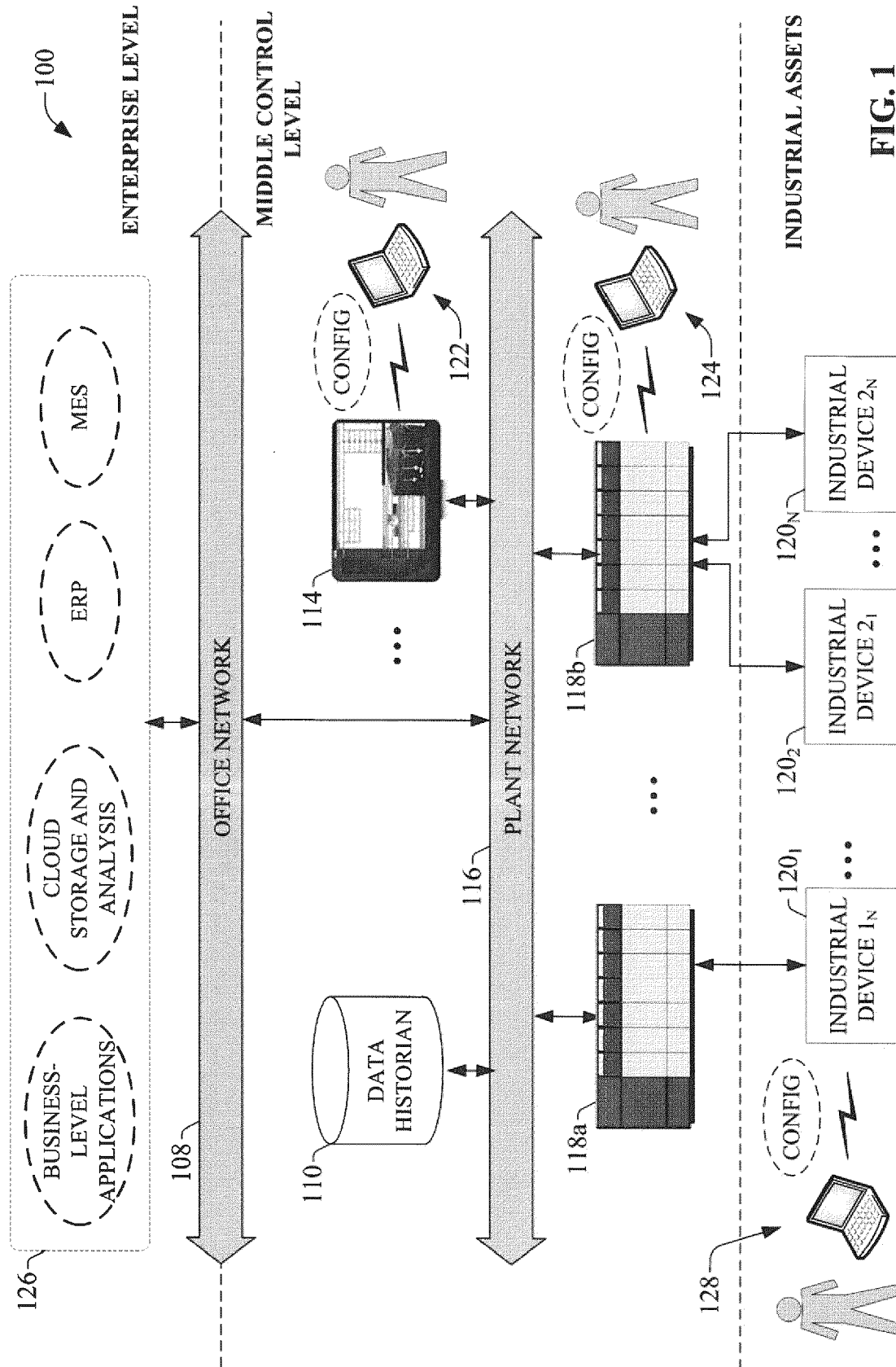
35

40

45

50

55



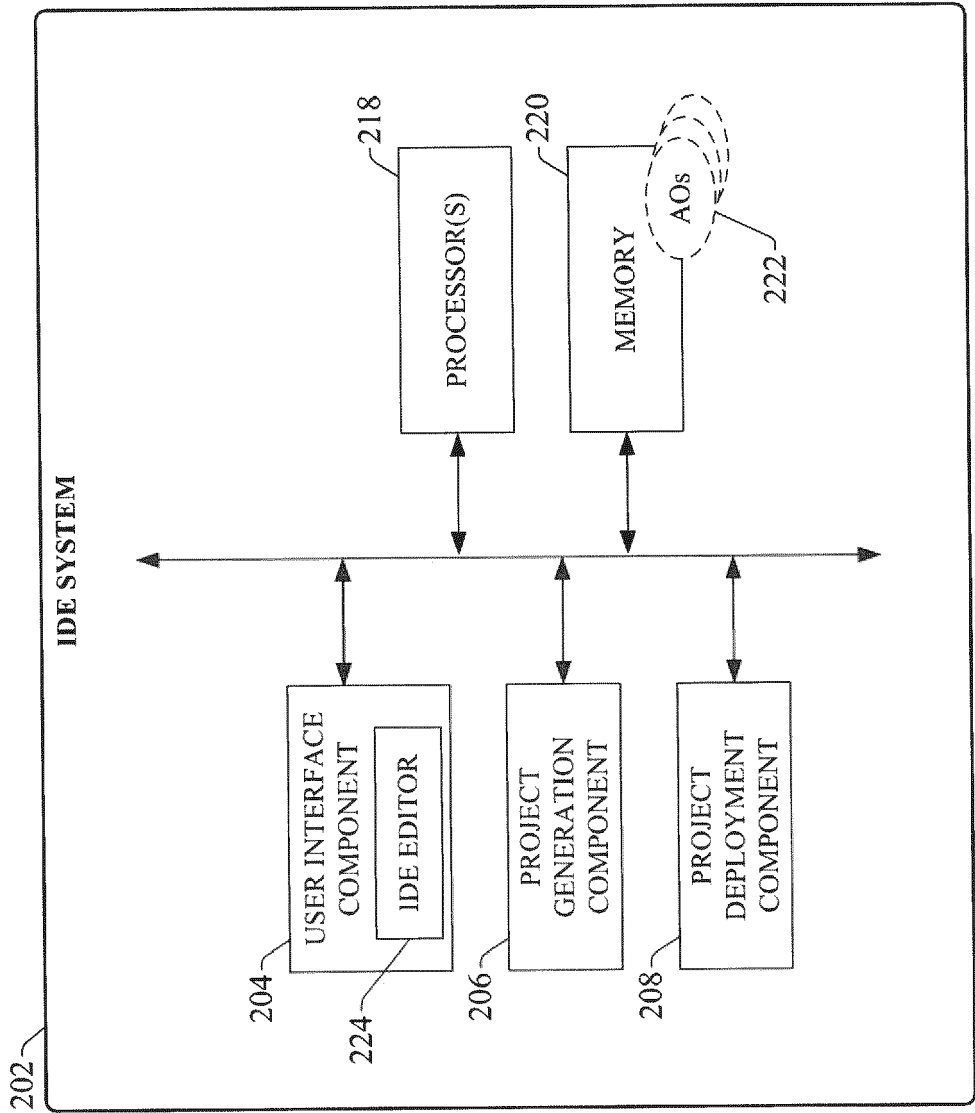


FIG. 2

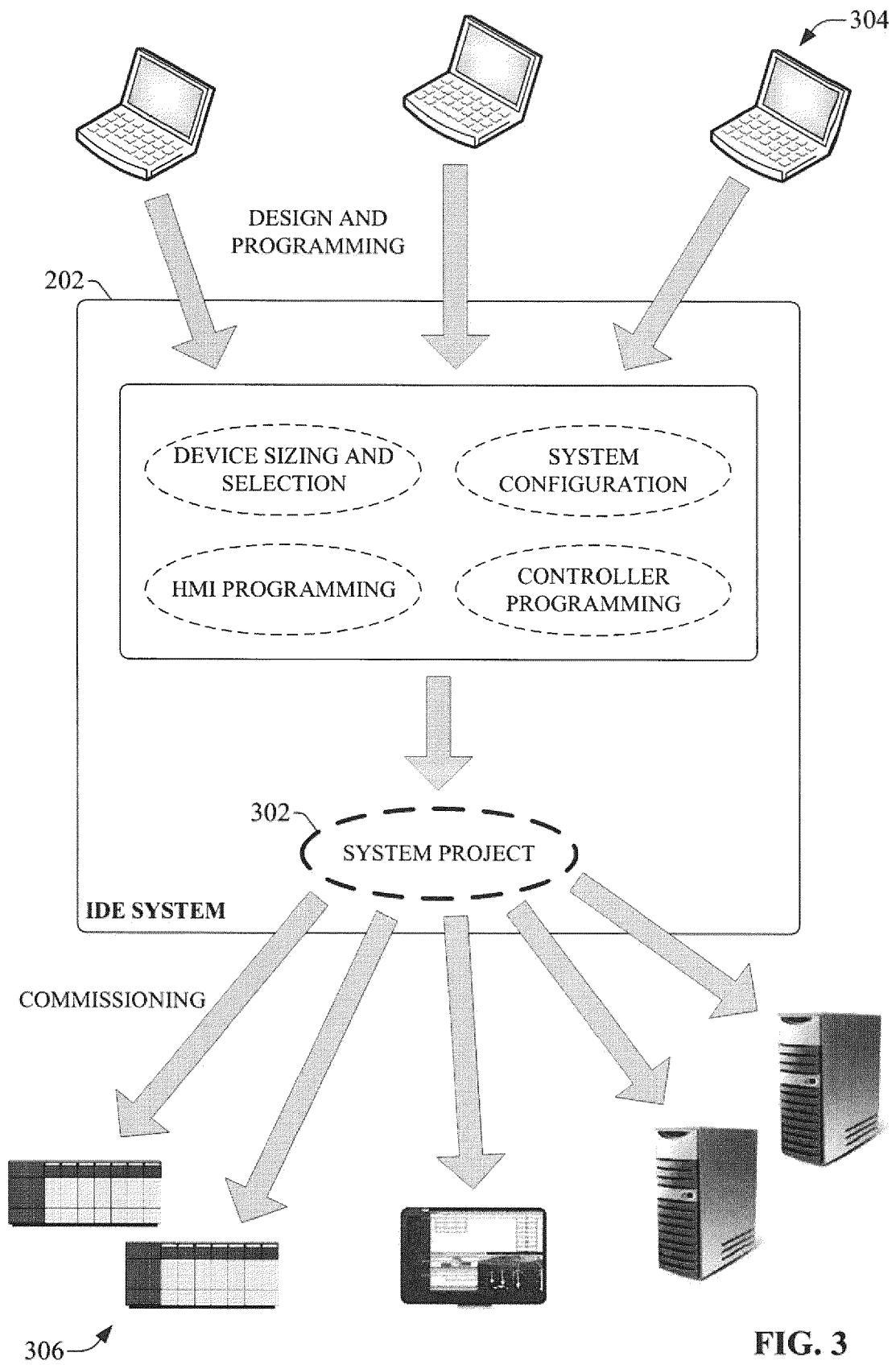


FIG. 3

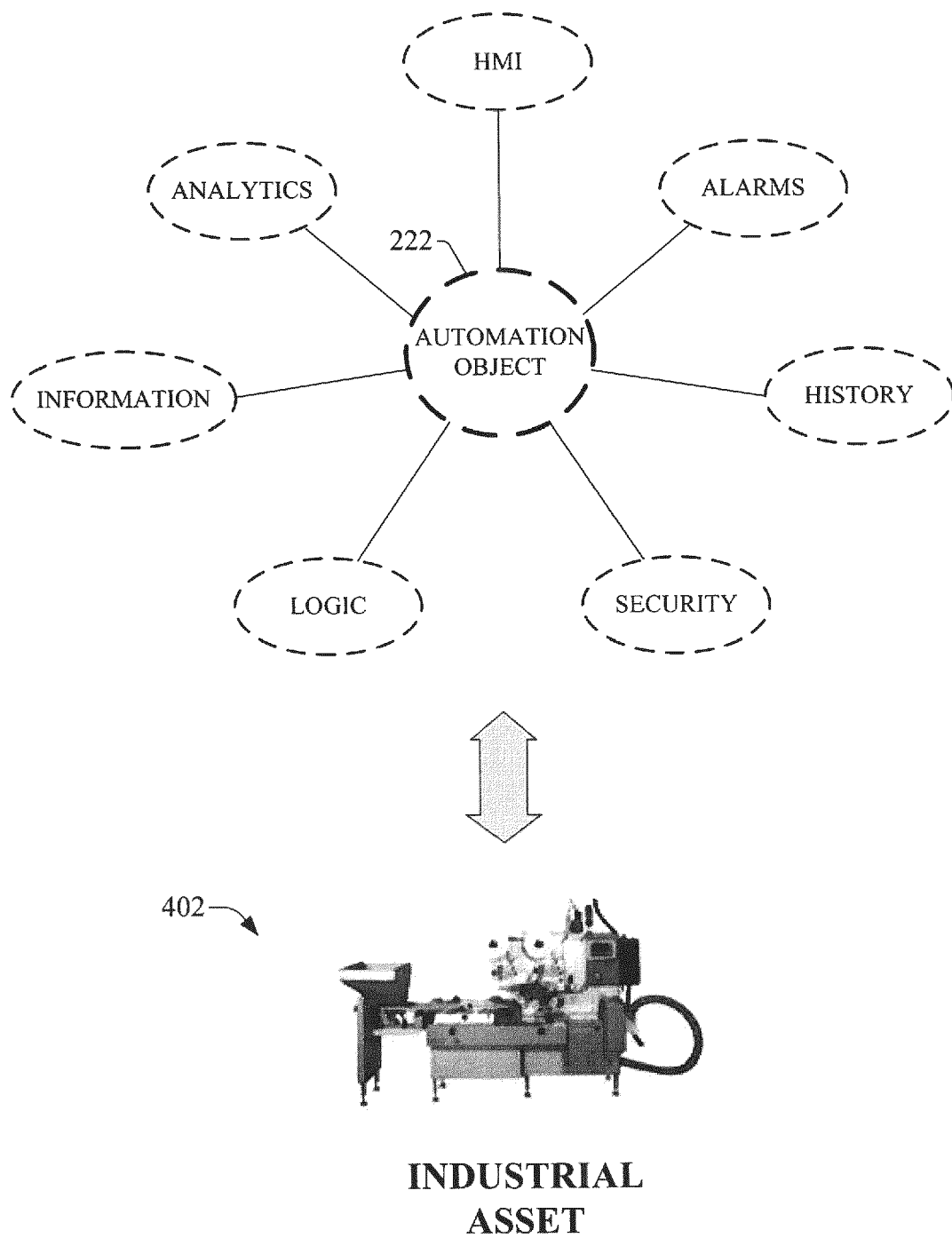


FIG. 4

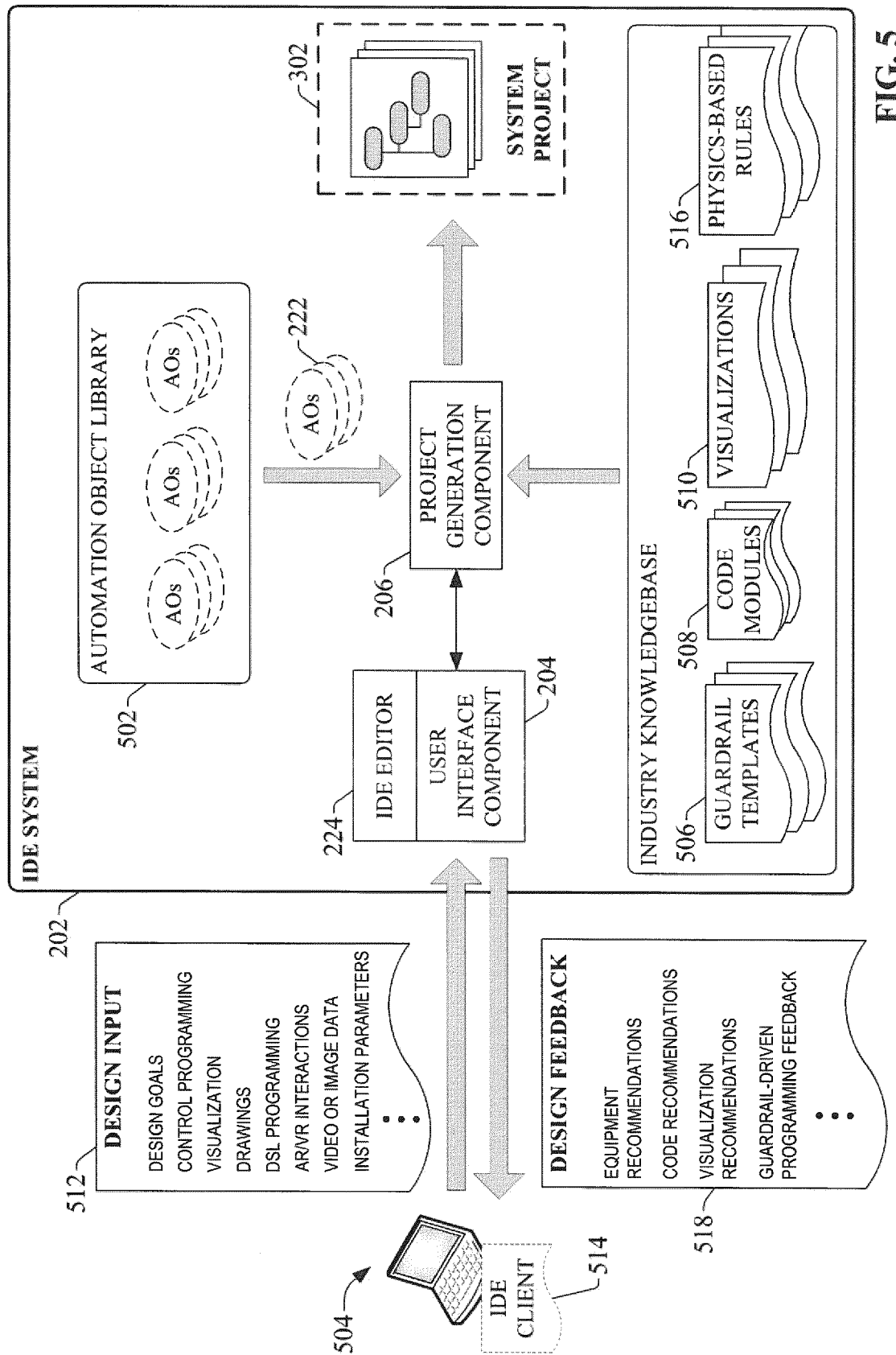


FIG. 5

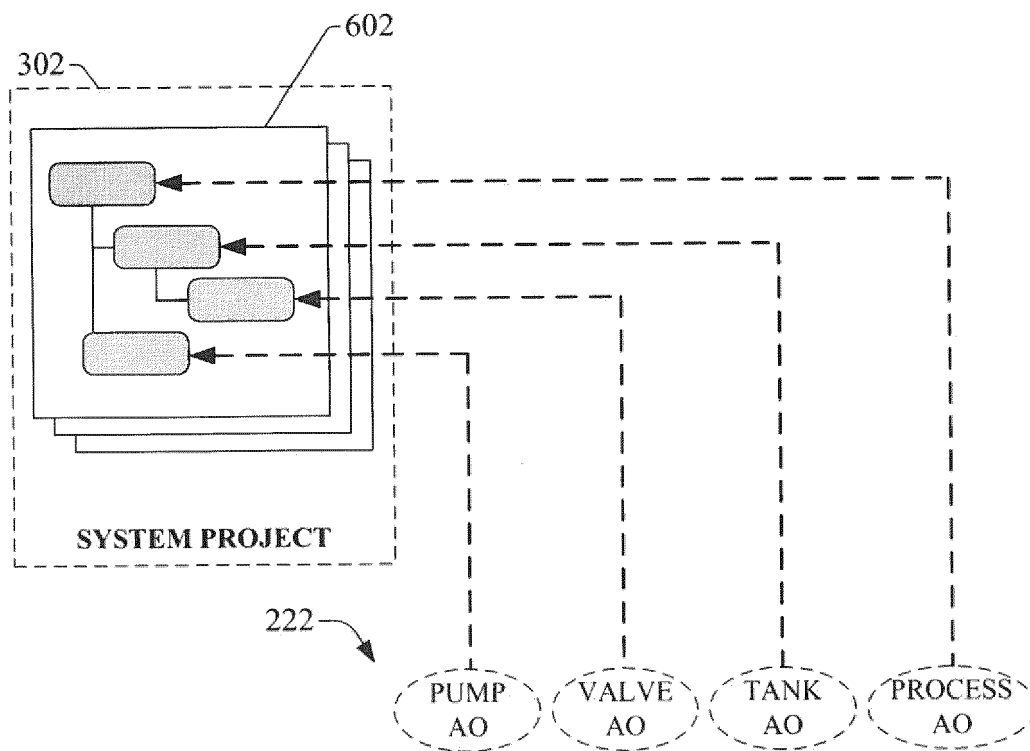


FIG. 6

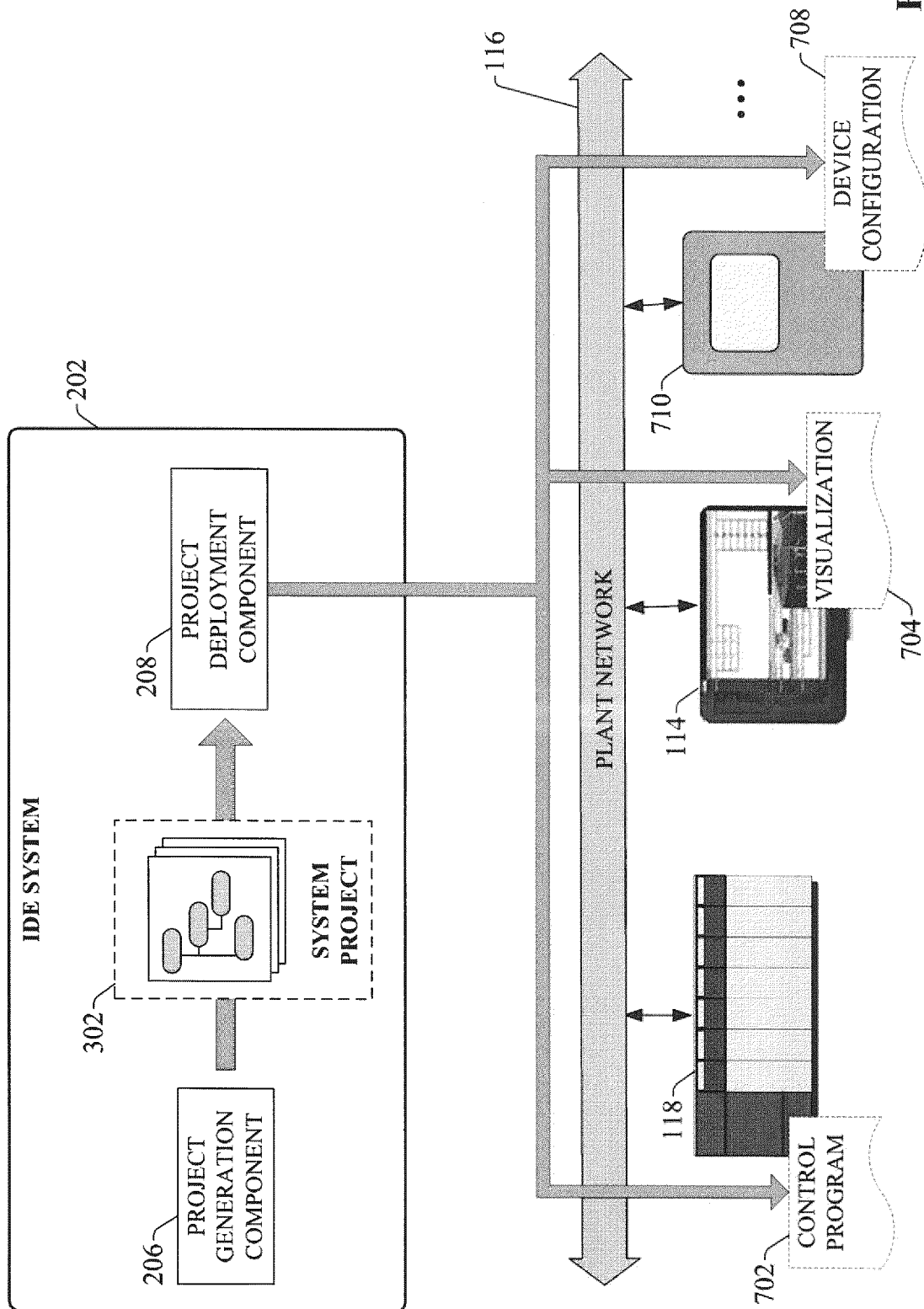


FIG. 7

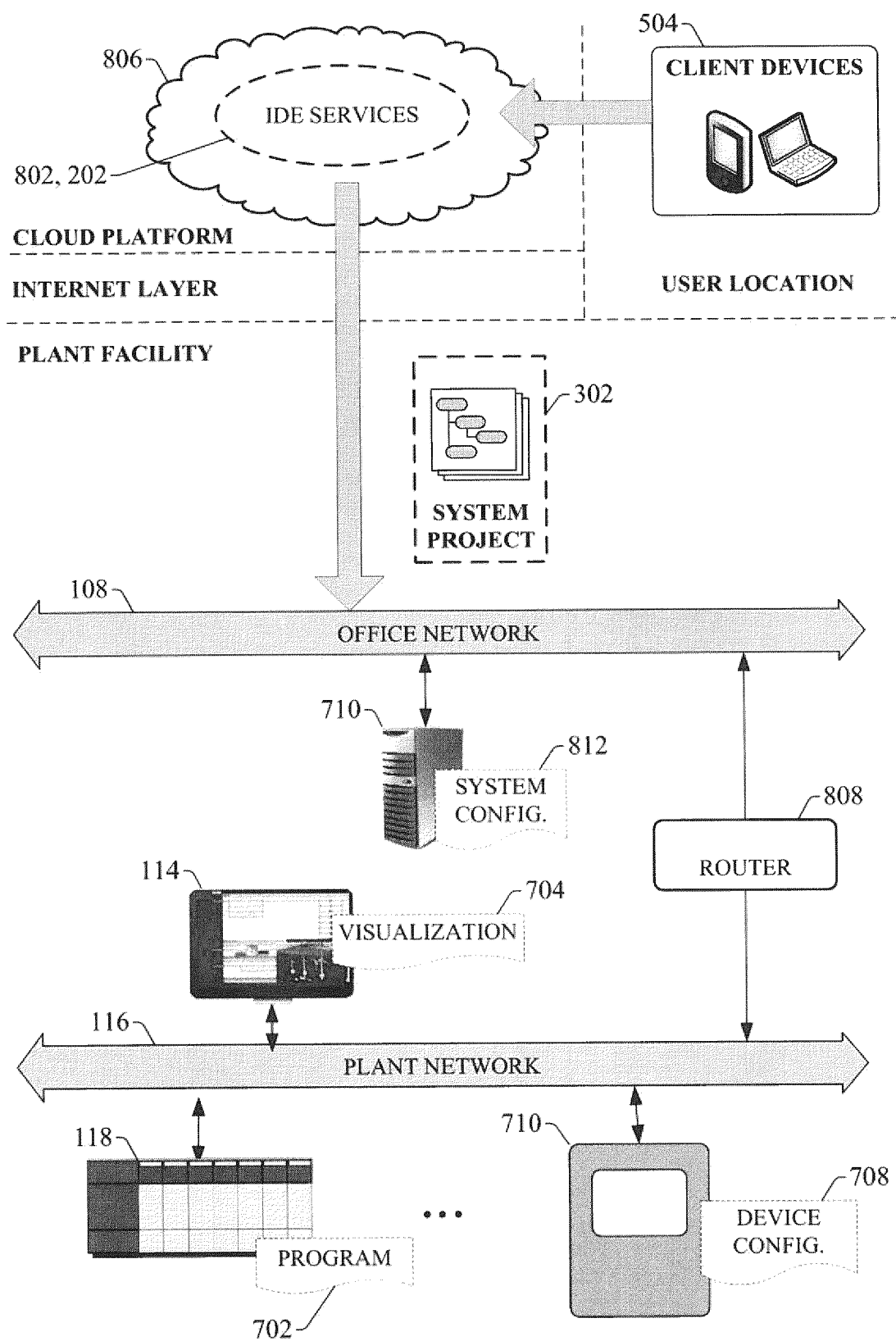


FIG. 8

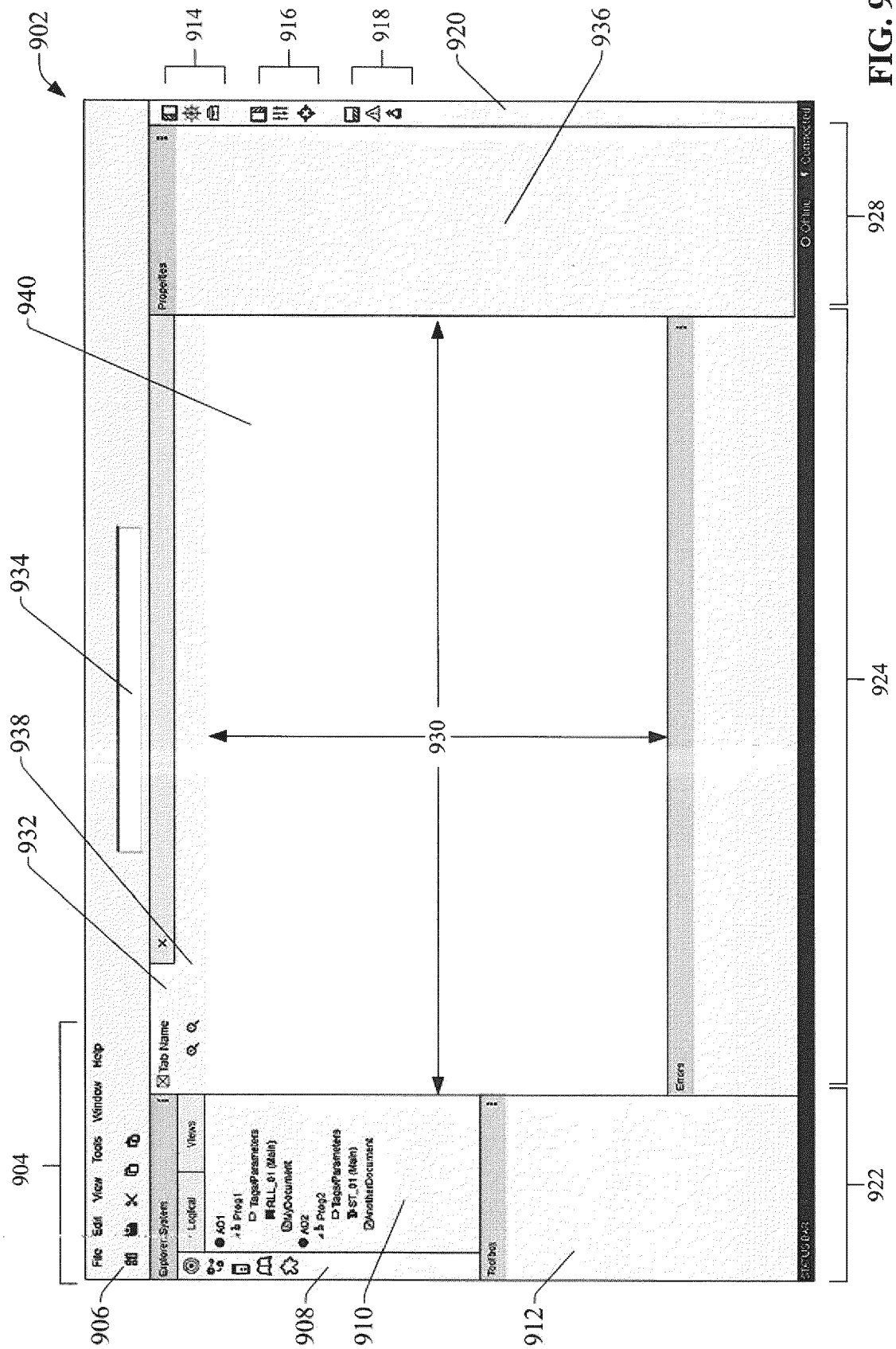


FIG. 9

1014

VIEW	
✓ Left Panels	ctrl + L
✓ Explorer	
Right Panels	
✓ Properties	
Online	
Cross Reference	
Bottom Panels	
Output	
Errors	
Current Content Panel	
✓ Panel One	
✓ Panel Two	

1016

FIG. 10b

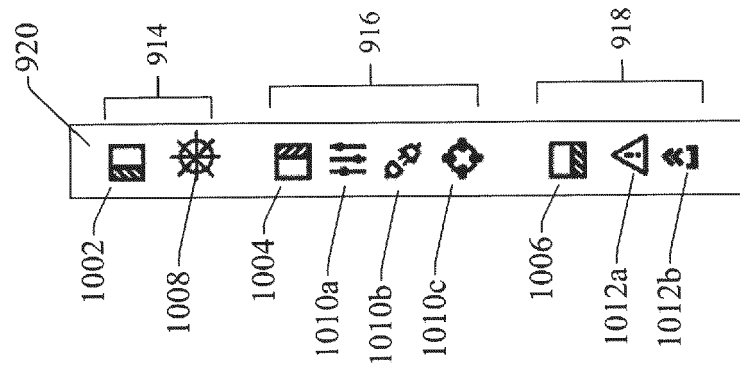
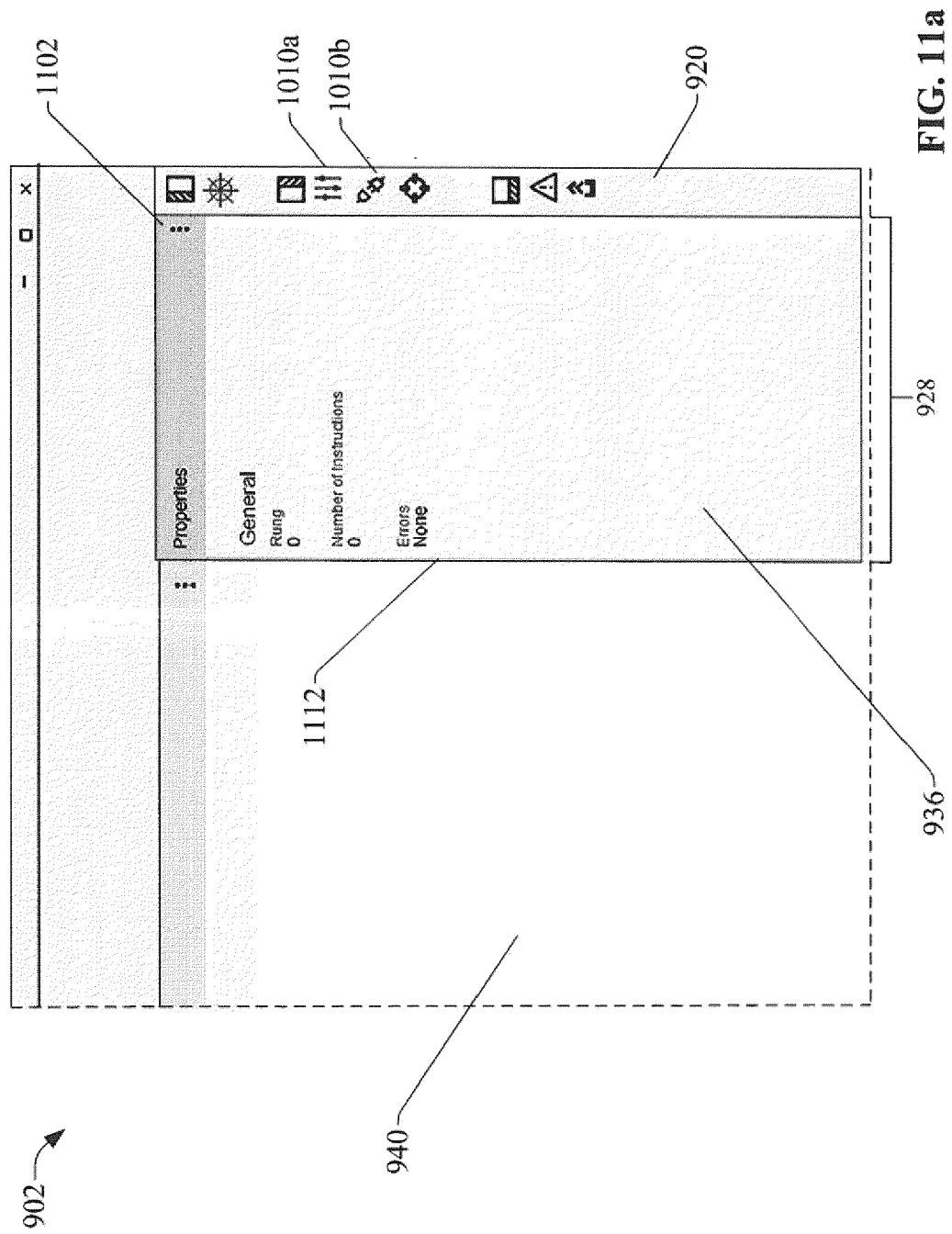


FIG. 10a



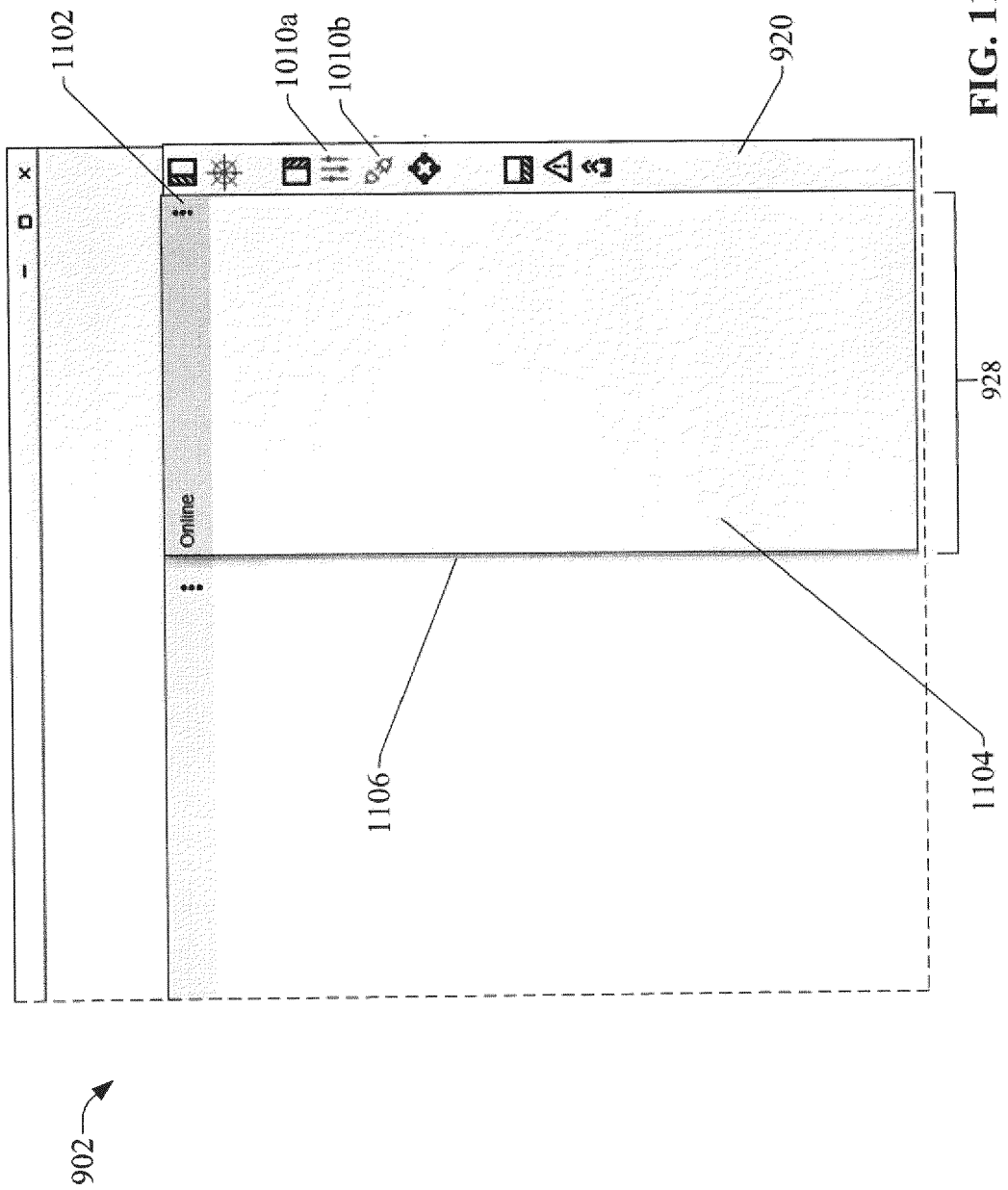


FIG. 11b

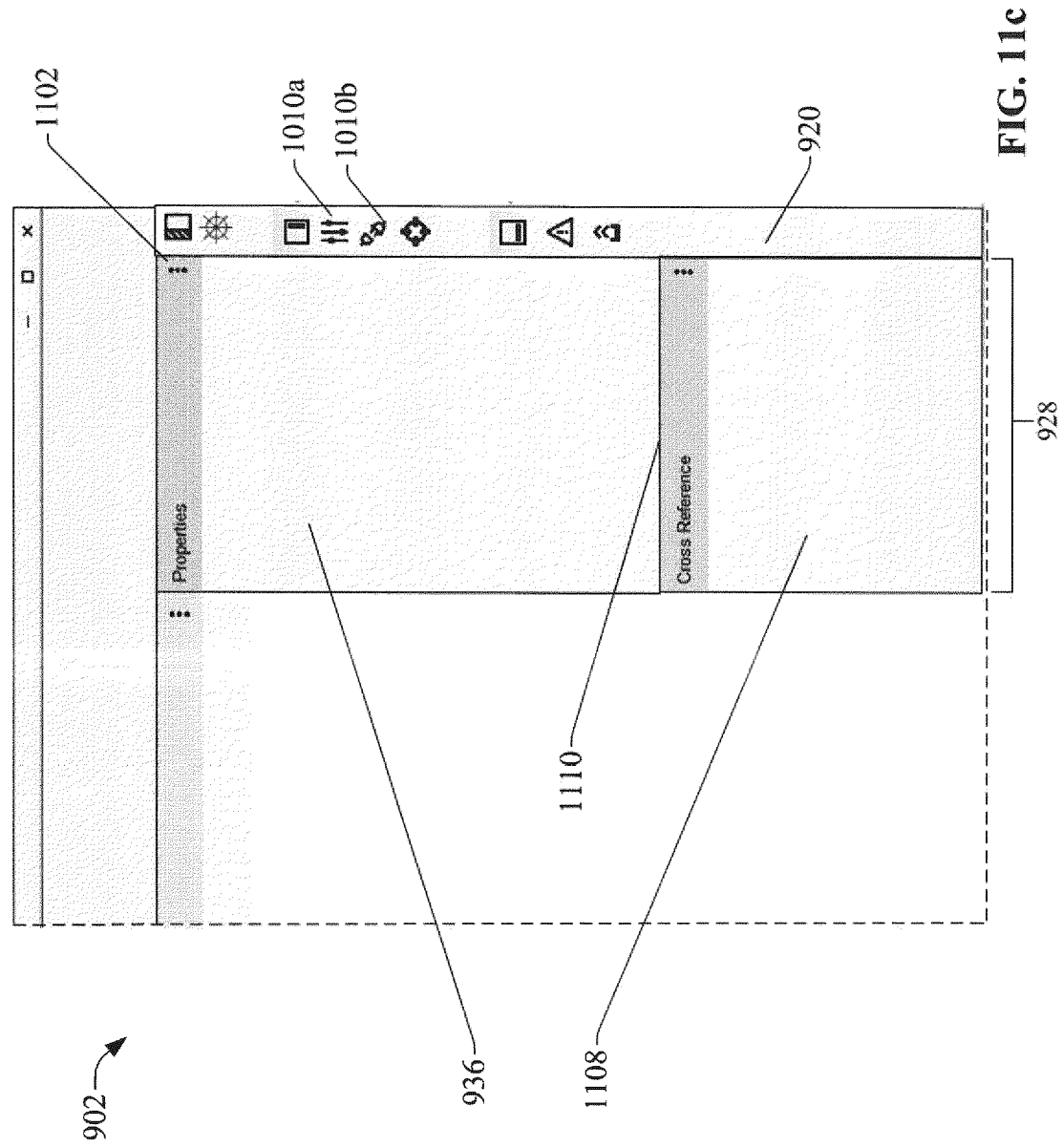


FIG. 11c

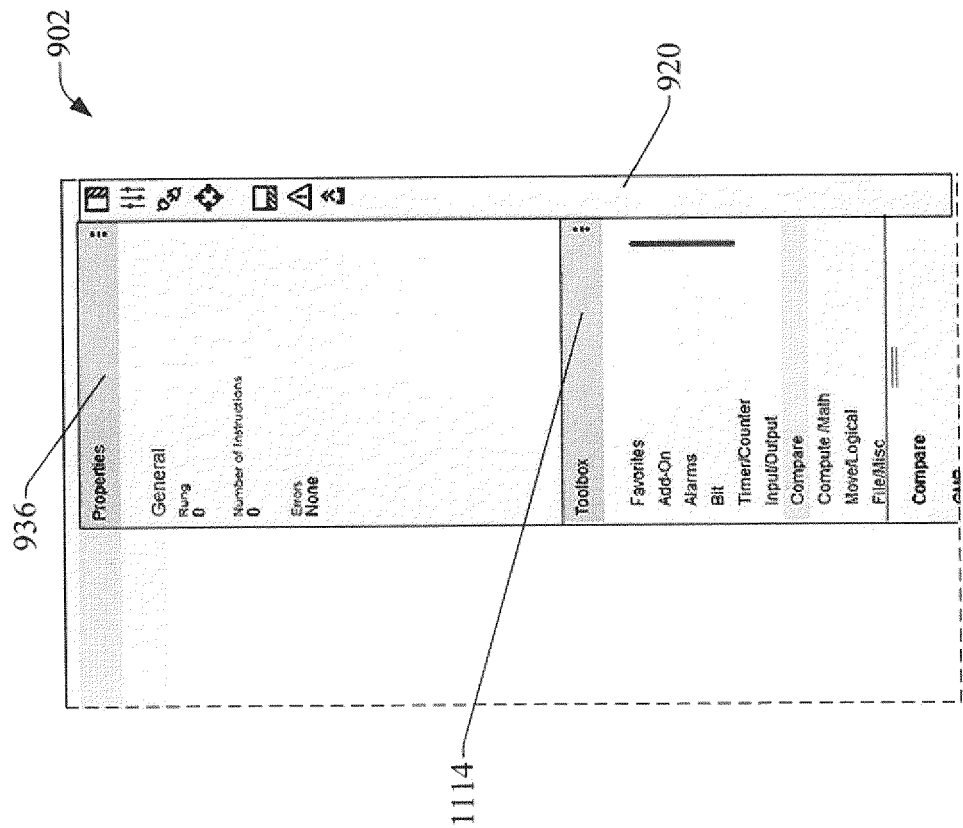


FIG. 11d

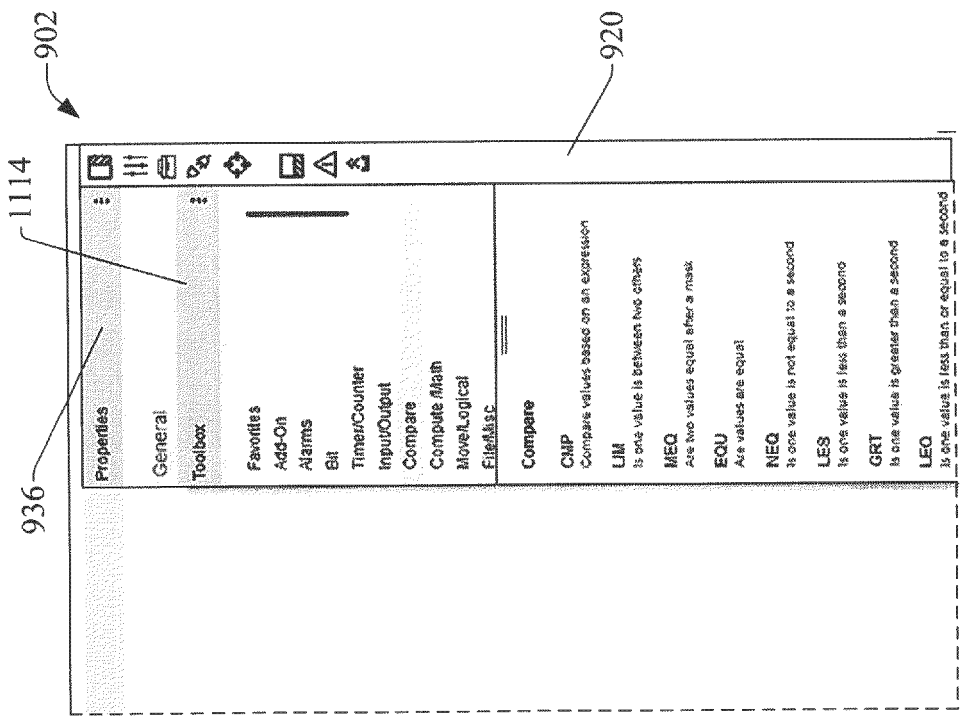


FIG. 11e

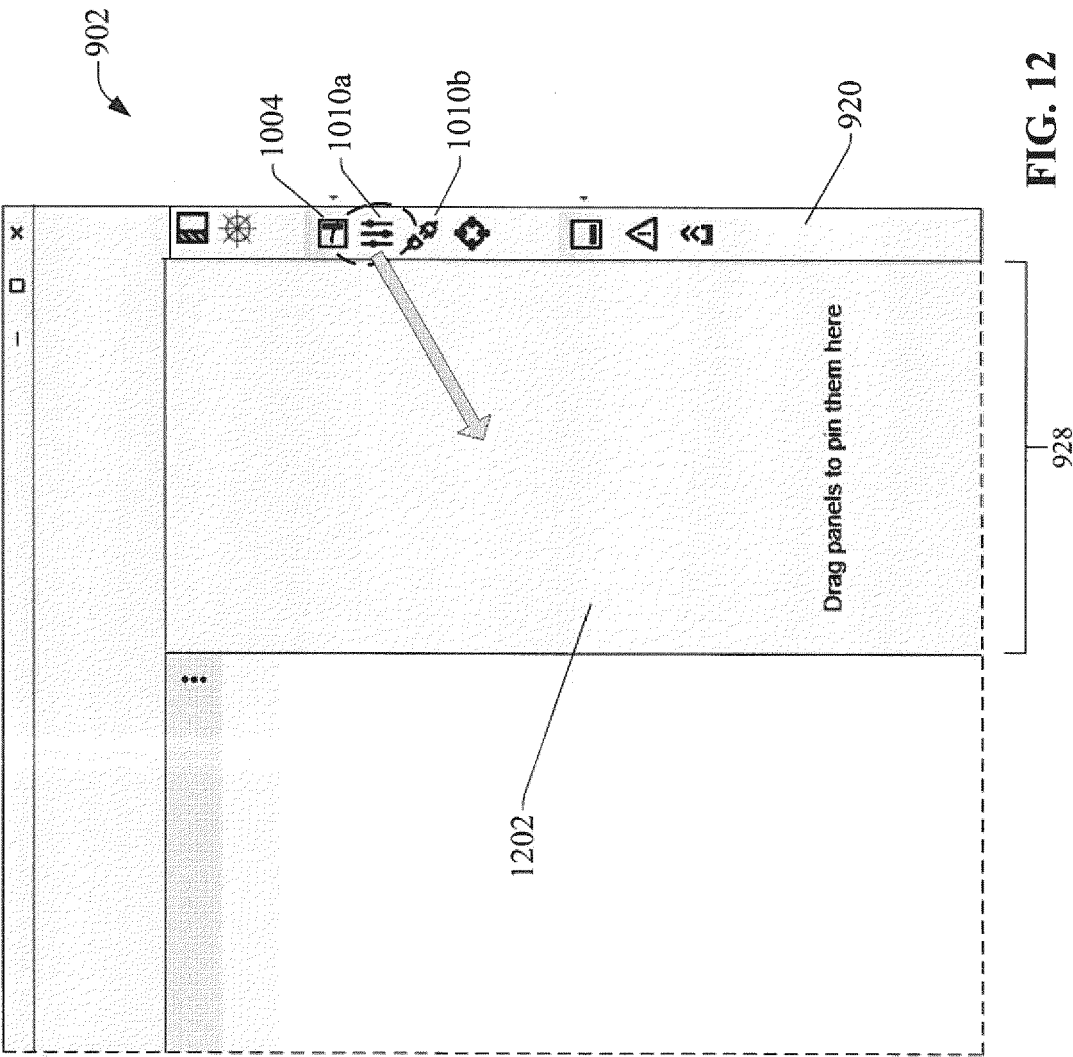


FIG. 12

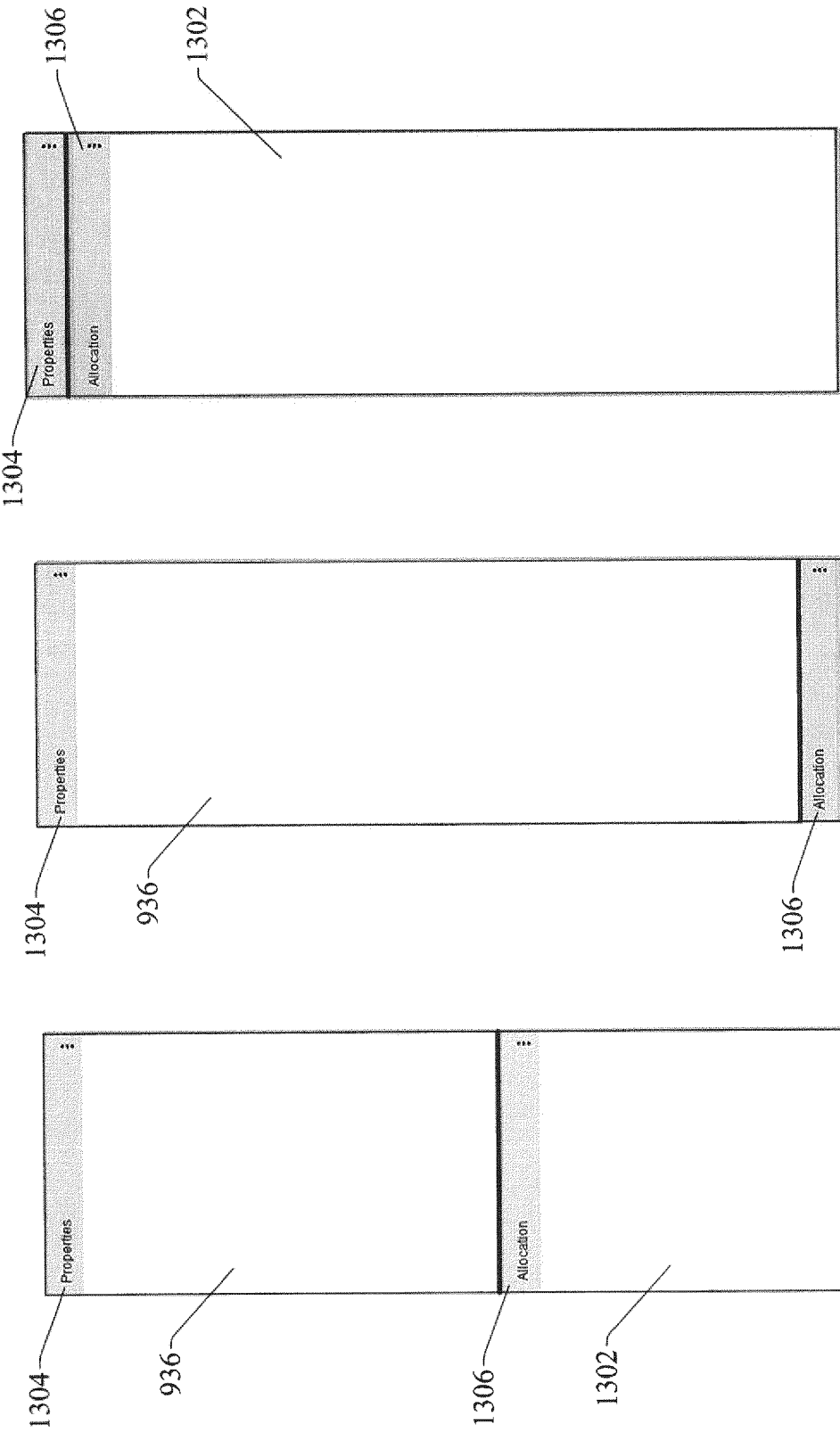


FIG. 13c

FIG. 13b

FIG. 13a

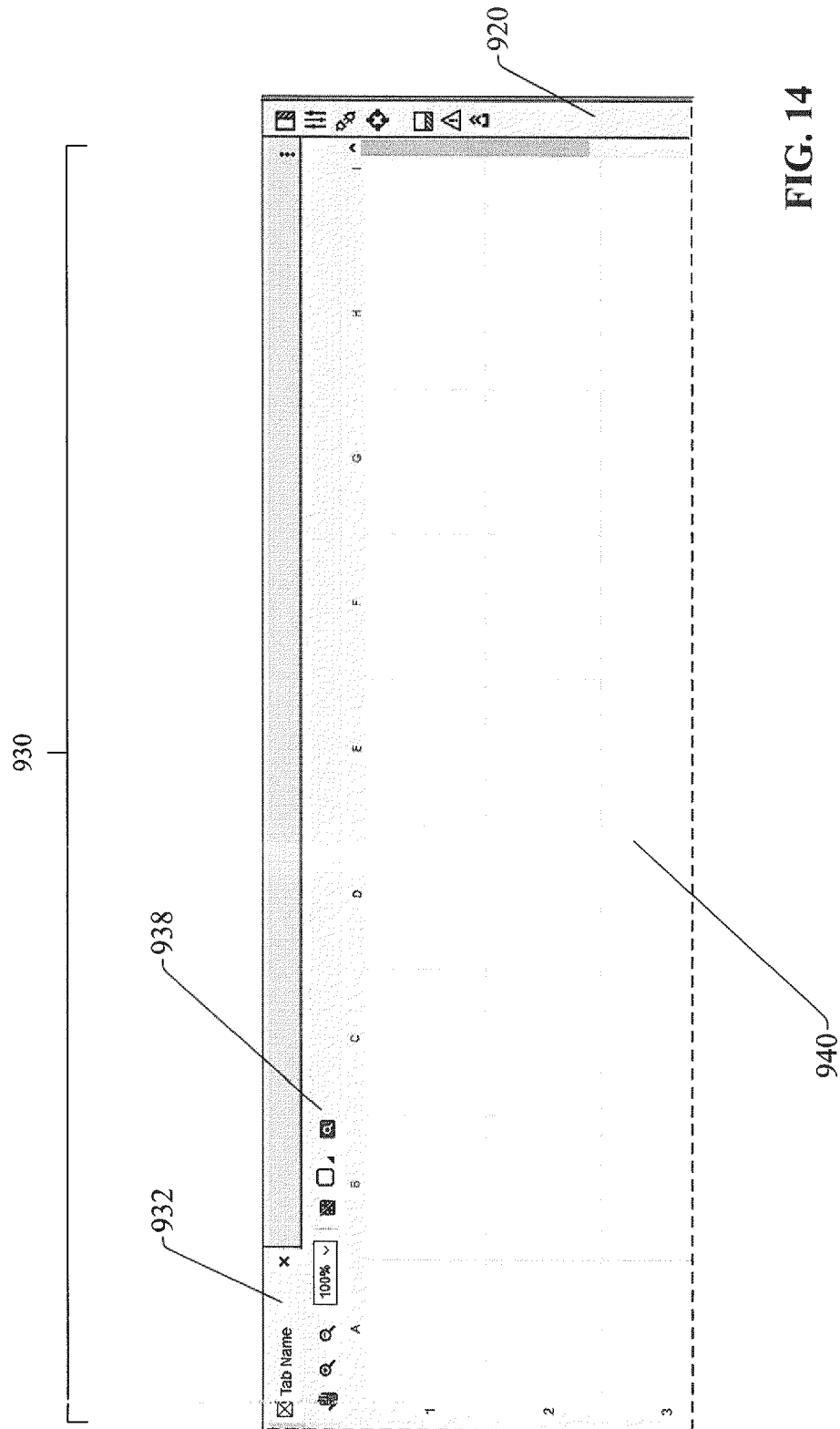


FIG. 14

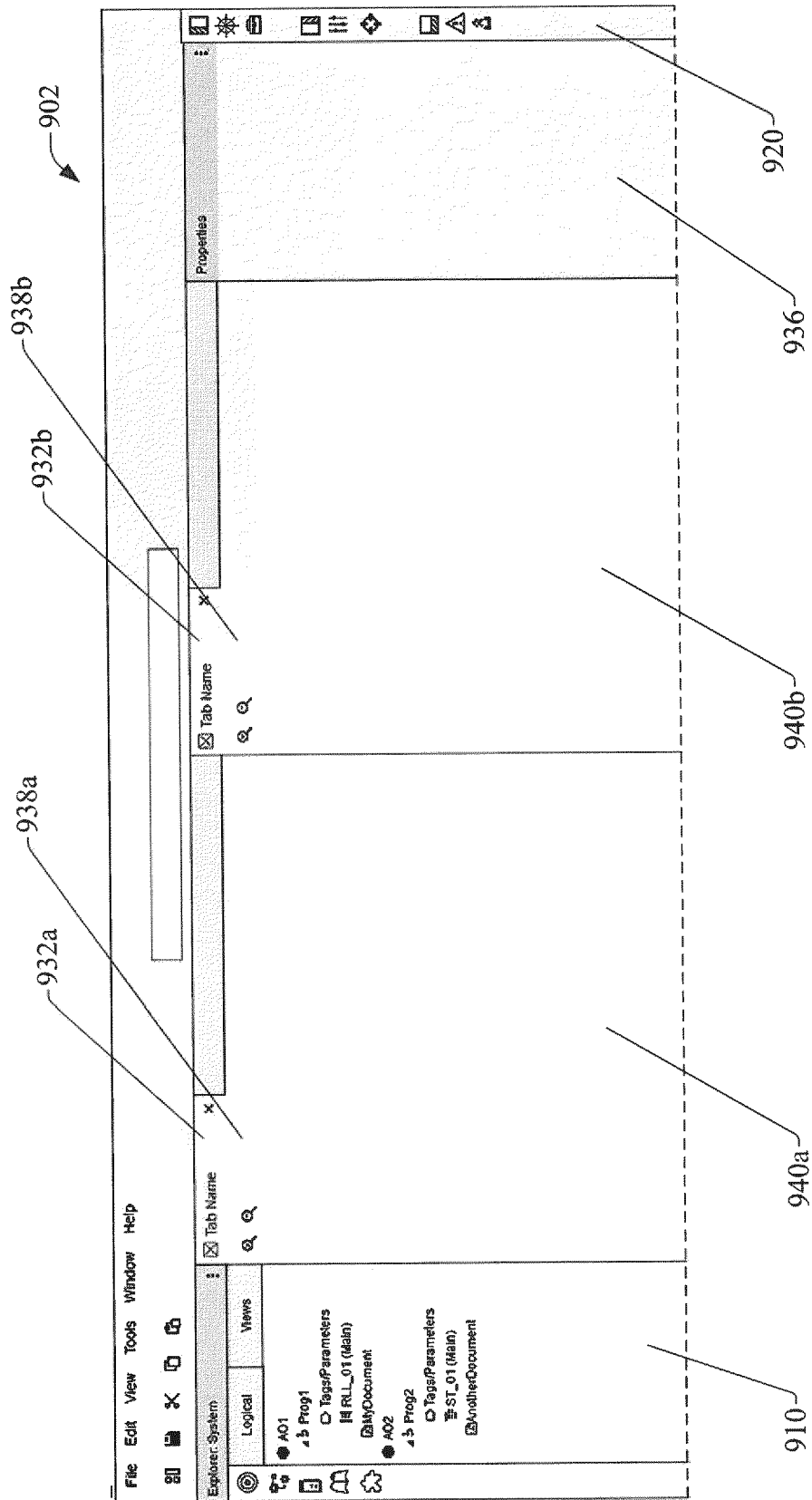


FIG. 15

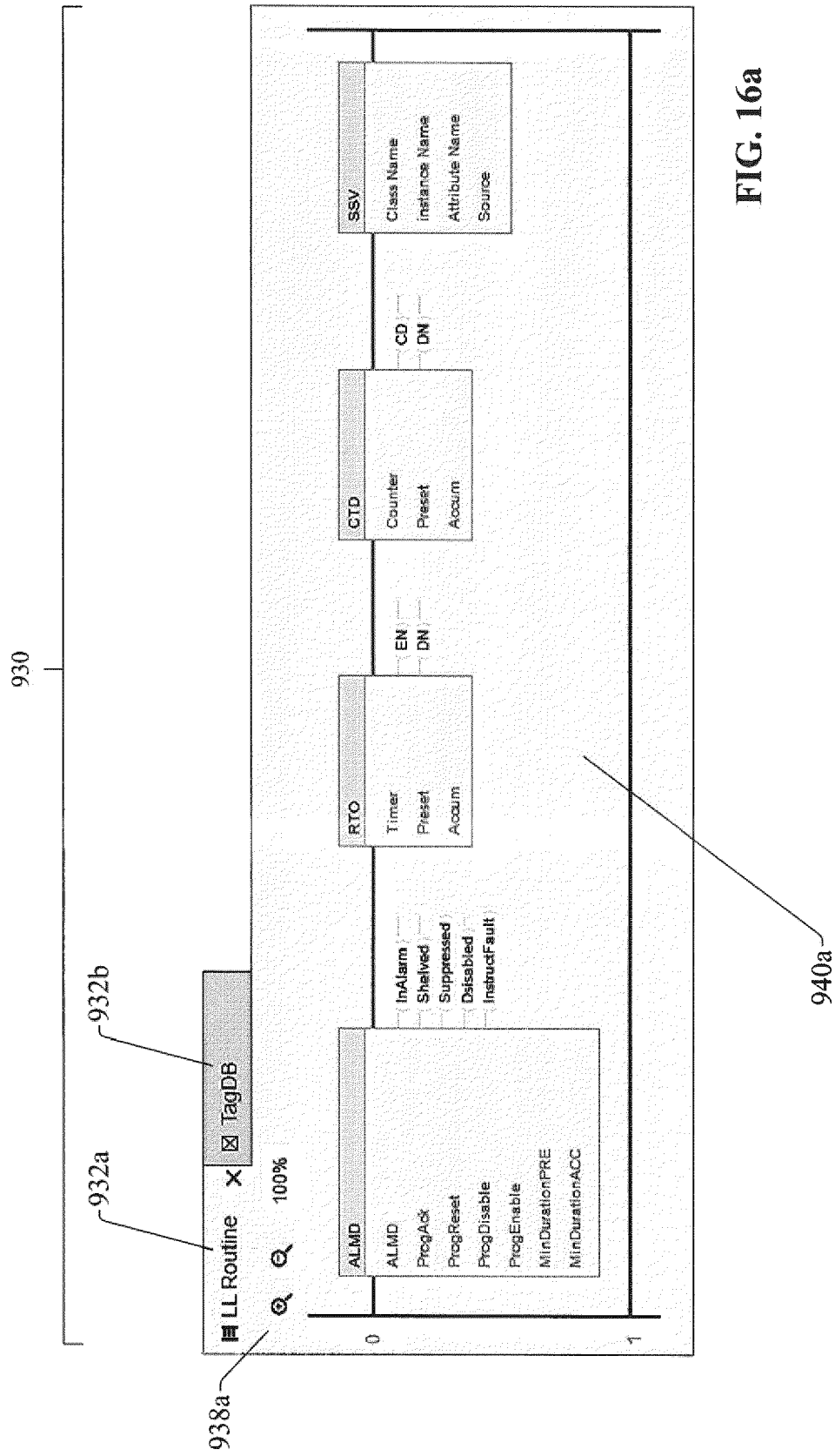


FIG. 16a

930

932a

932b

940b

λ FBD Routine X

☒ TagDB

Name	Default	Style	Data Type	History	Alarms	Description
BNOT	0	Decimal	FBD_BOOLEAN			
CTUD	0	Decimal	FBD_COUNTER			
TONR	0	Decimal	FBD_TIMER			
TOT	0	Decimal	TOTALIZER			

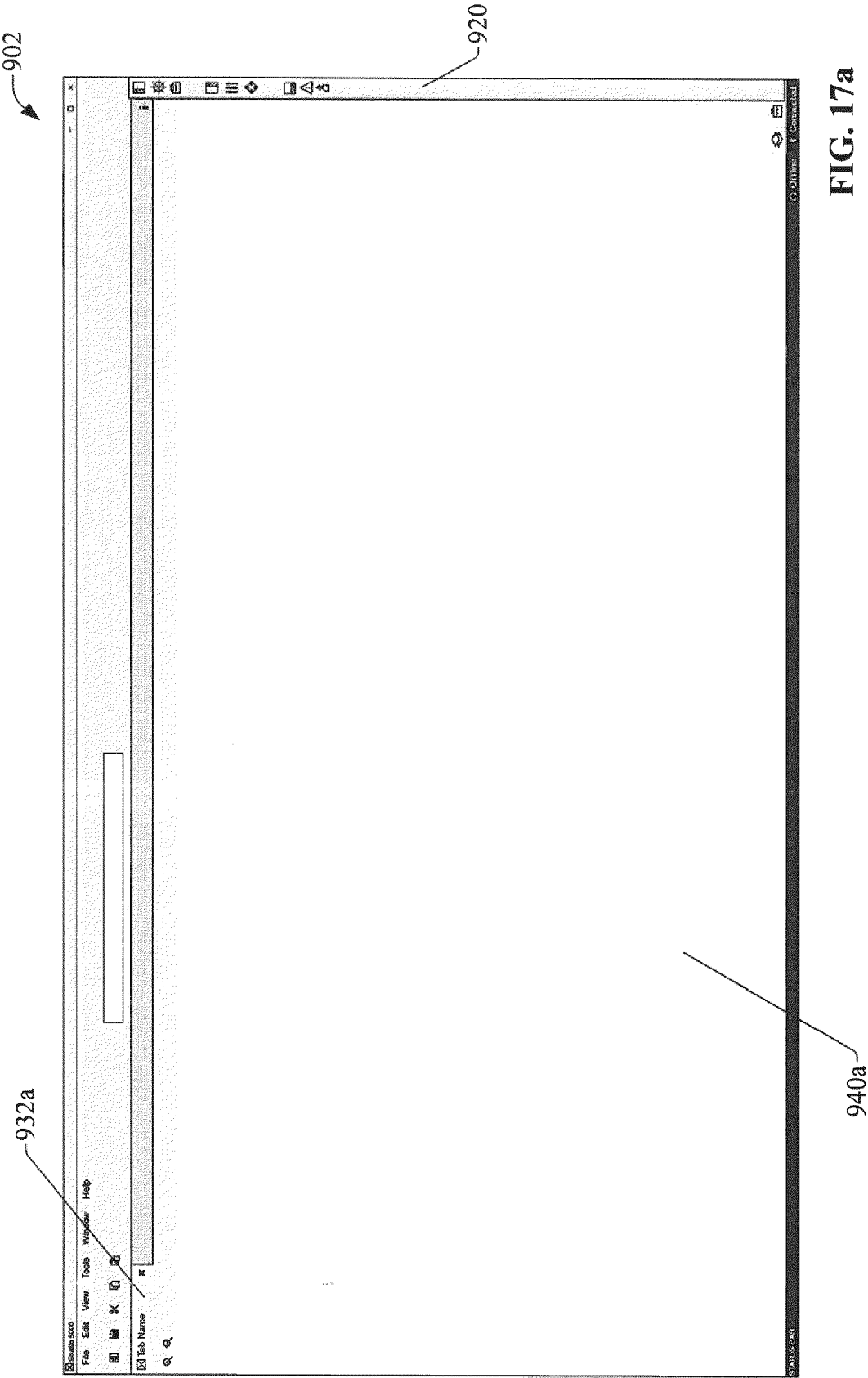
Parameters

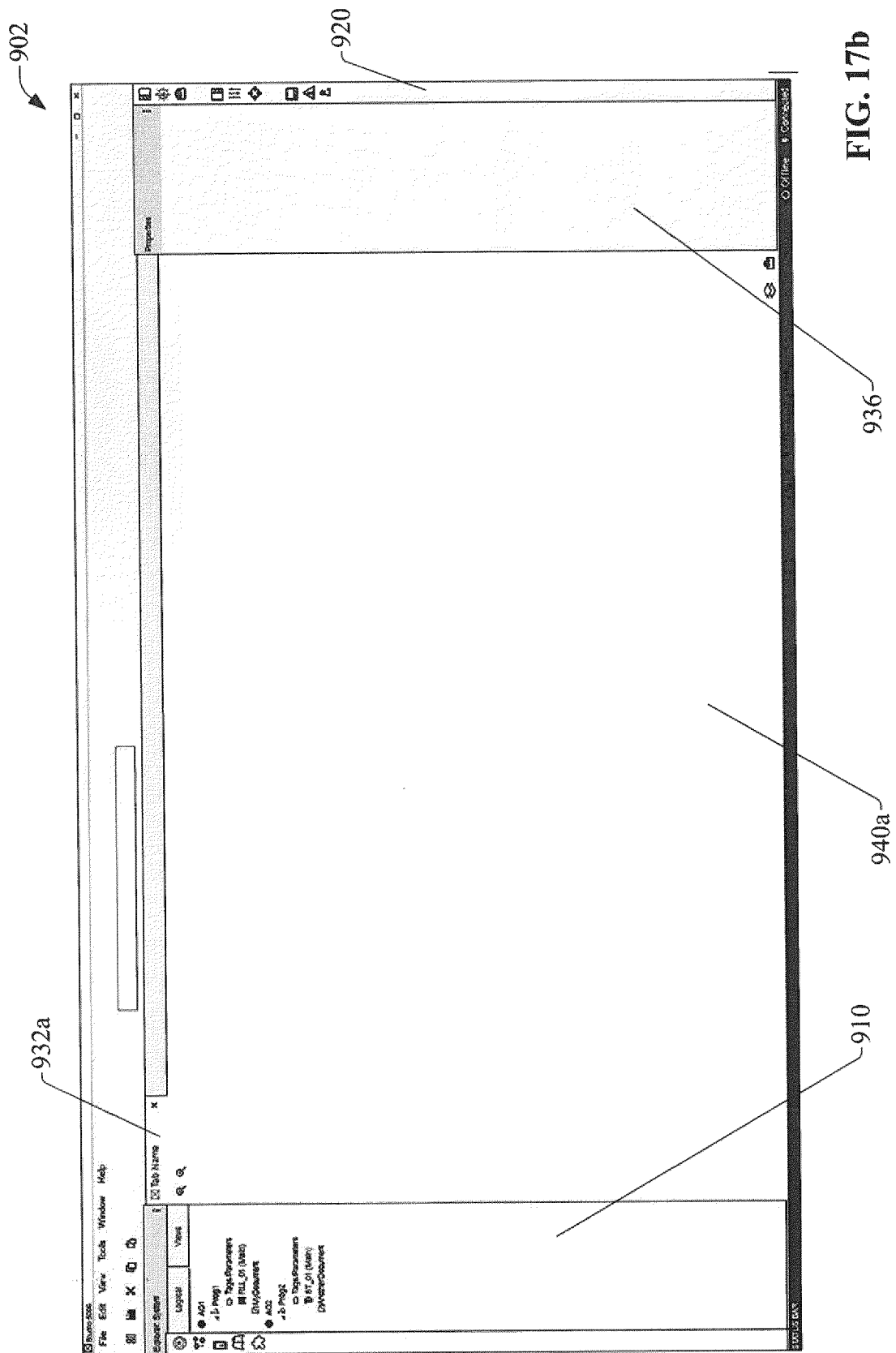
Name	Default	Style	Usage	Data Type	History	Alarms	Description
Out_Total	0	Decimal	Output	REAL			
Alm_InFault	0	Decimal	Output	BOOL			
Inp_PV	0	Decimal	Input	BOOL			
Inp_ResetValve	0.0	Float	Input	REAL			
Cfg_Target	0.0	Float	Input	REAL			
Cfg_TargetDev1	0.0	Float	Input	REAL			

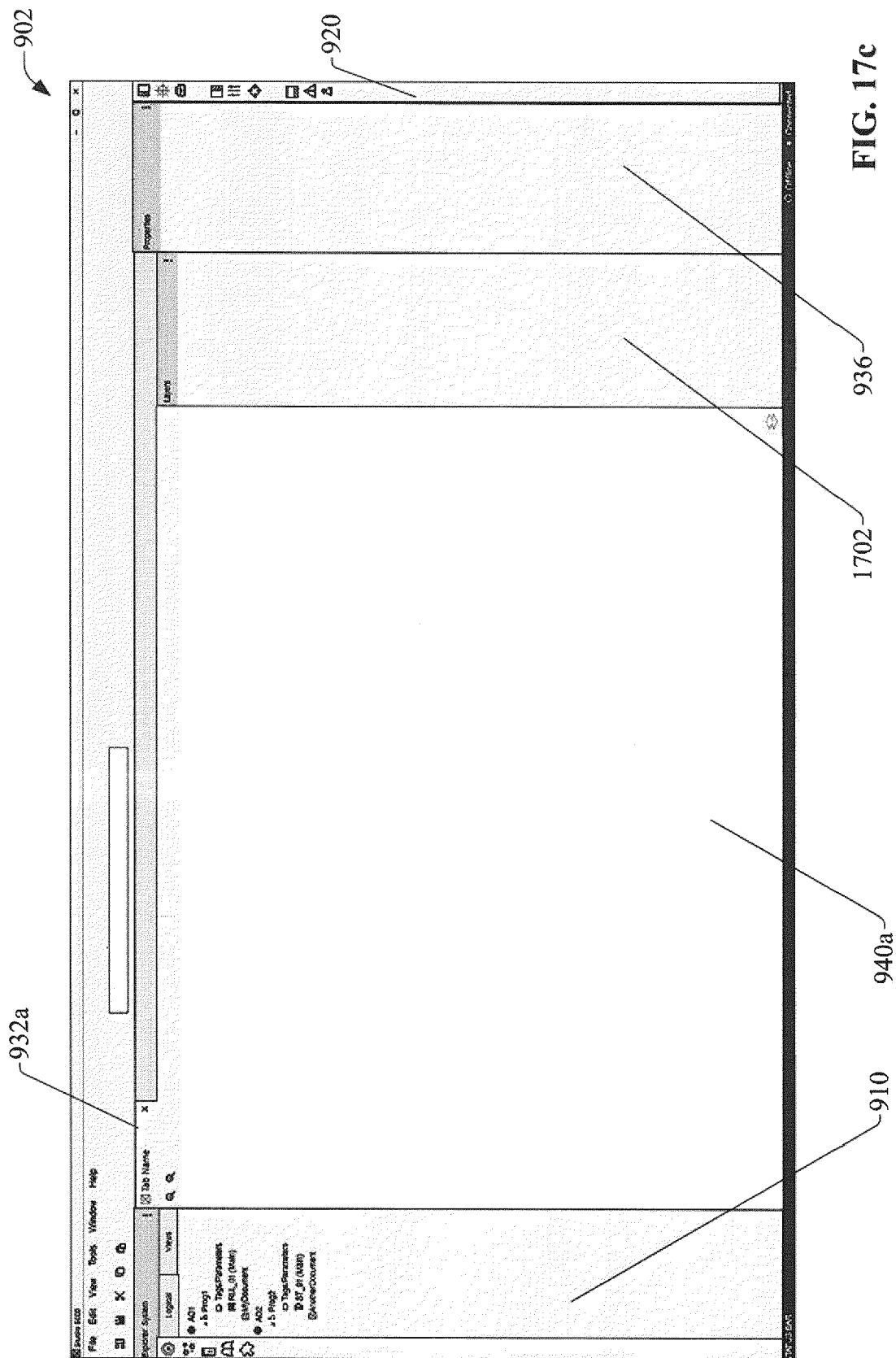
Import

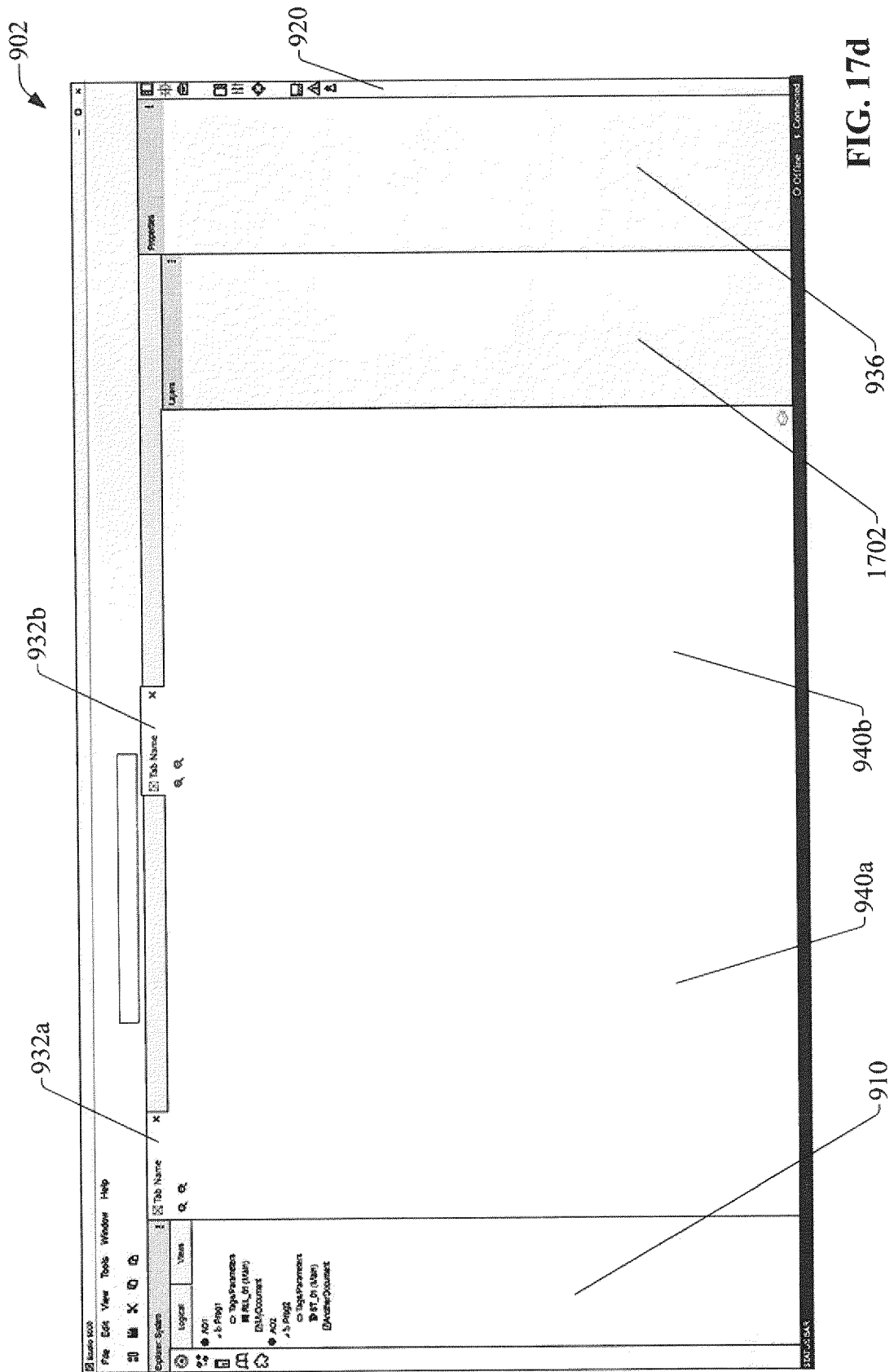
Import

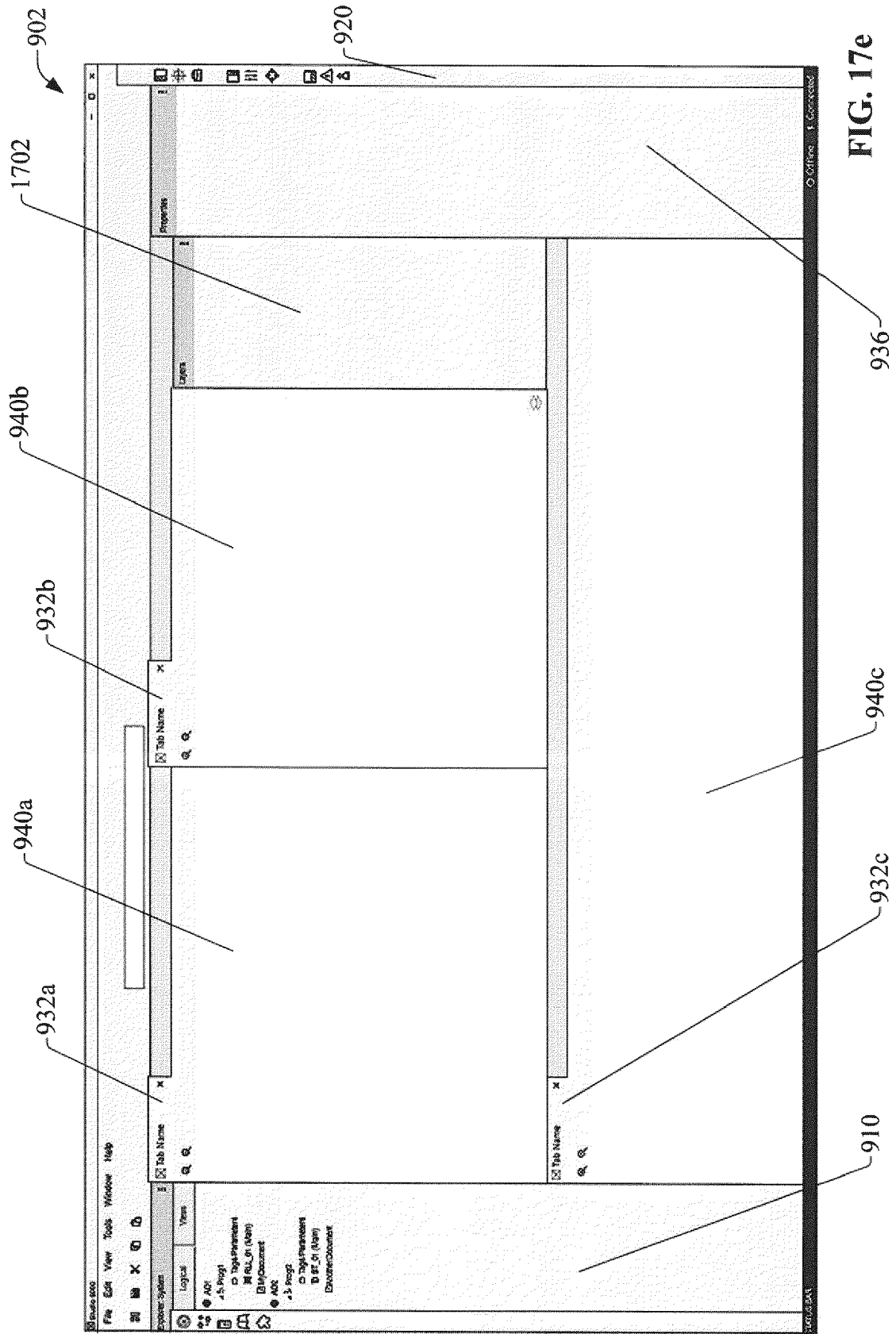
FIG. 16b

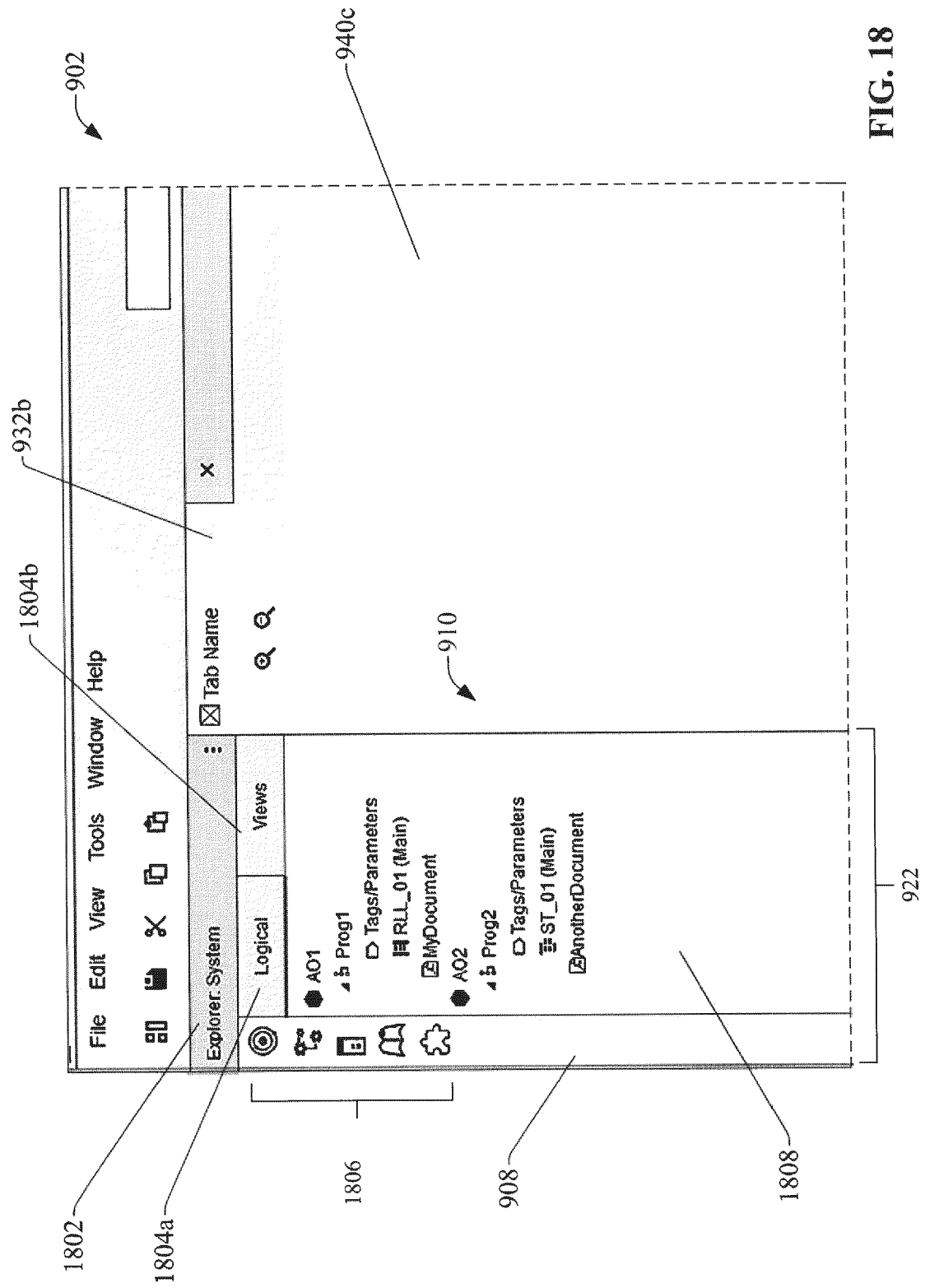












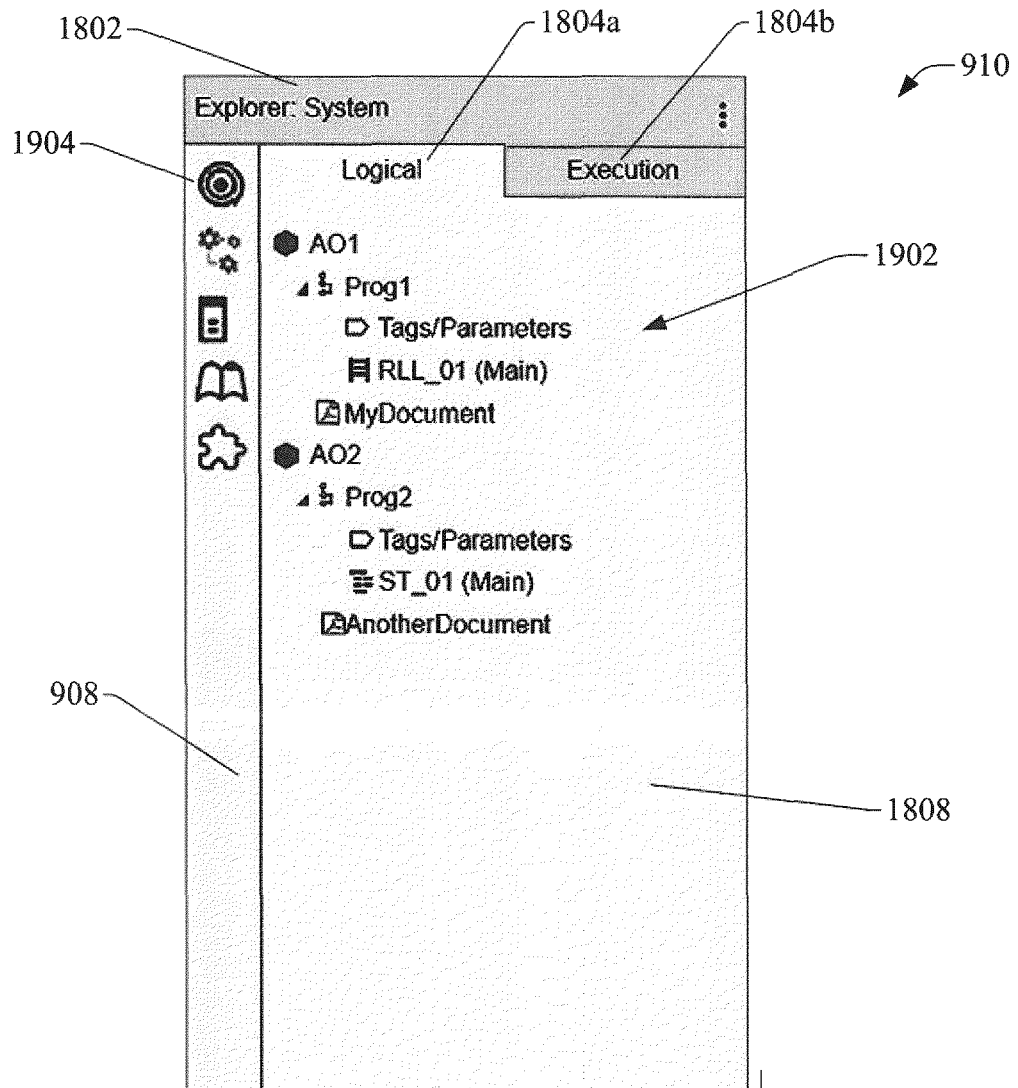


FIG. 19a

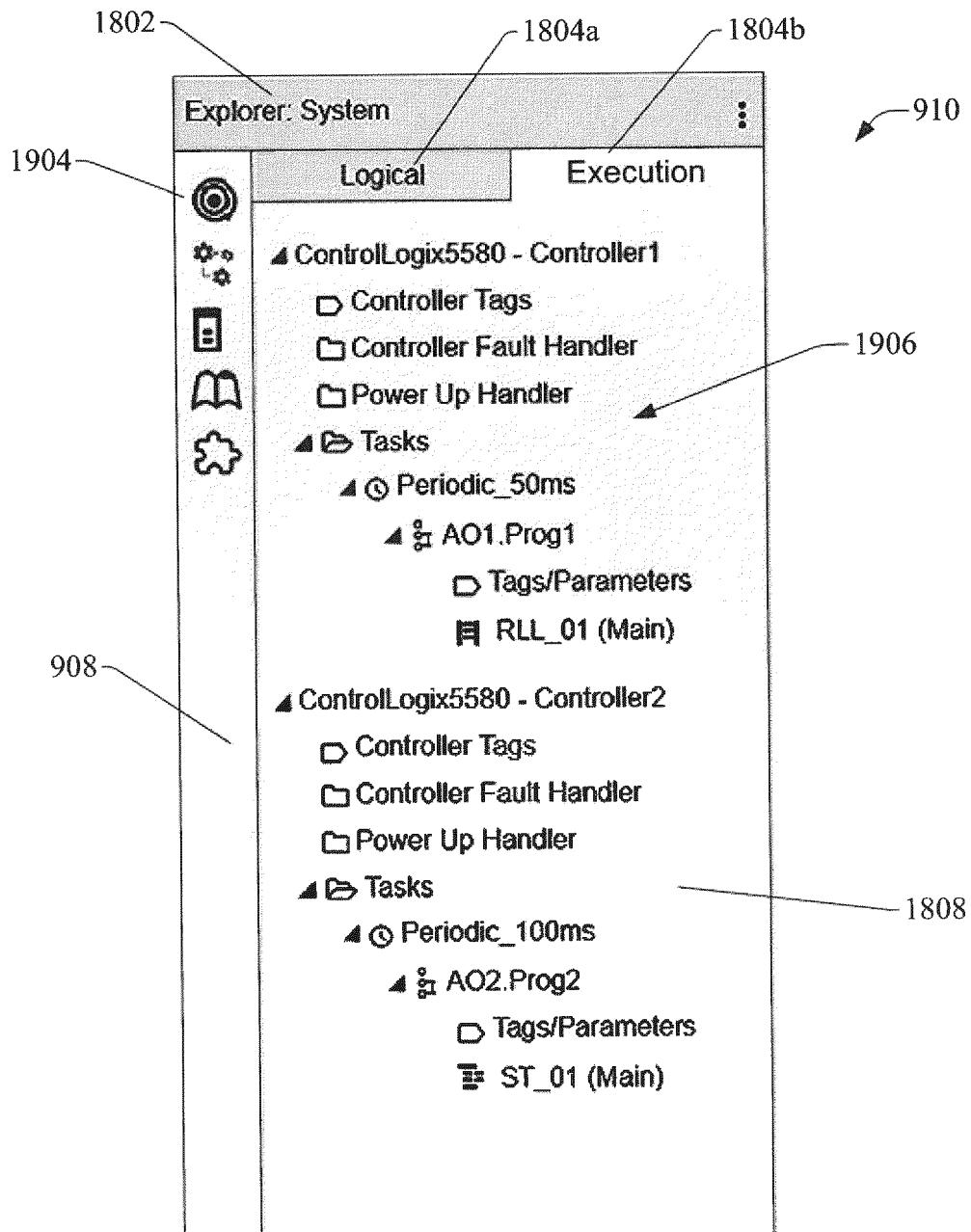


FIG. 19b

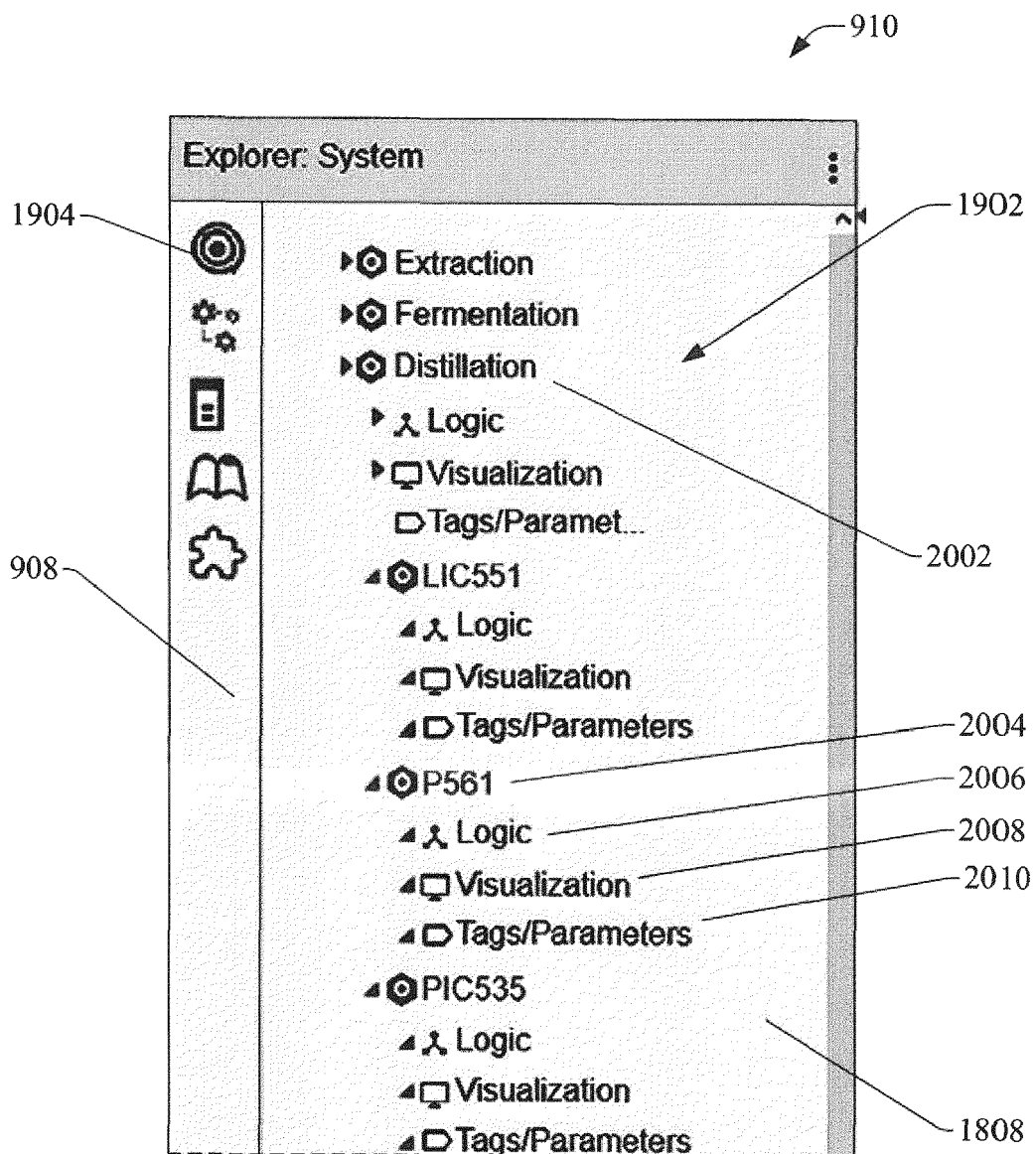


FIG. 20

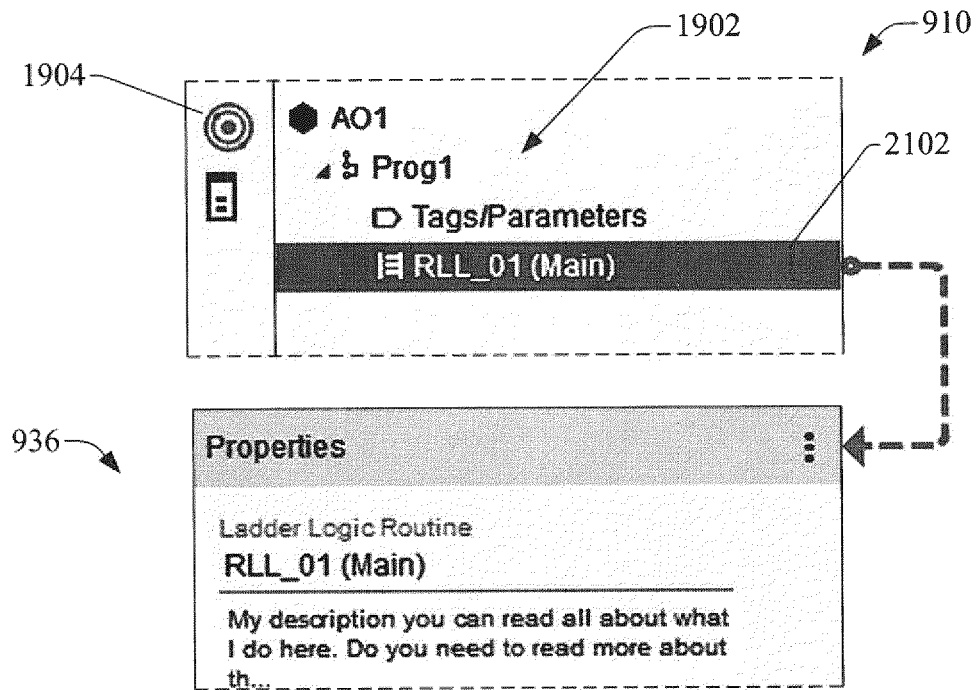


FIG. 21a

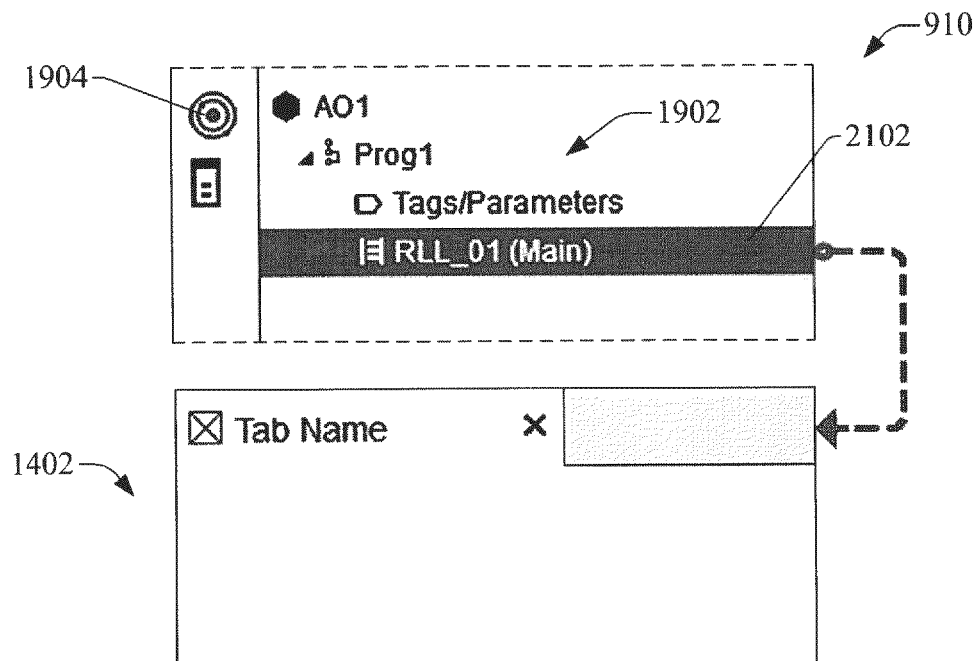


FIG. 21b

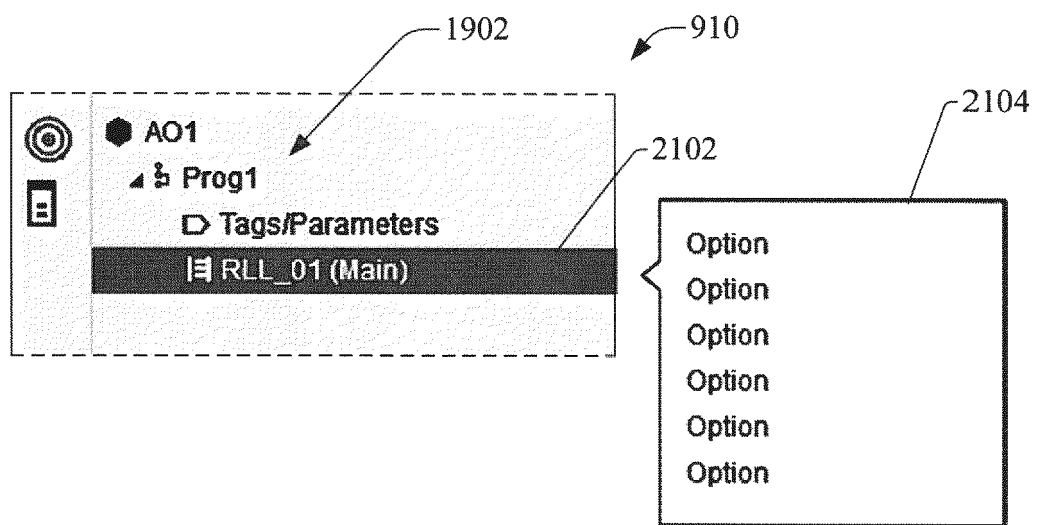


FIG. 21c

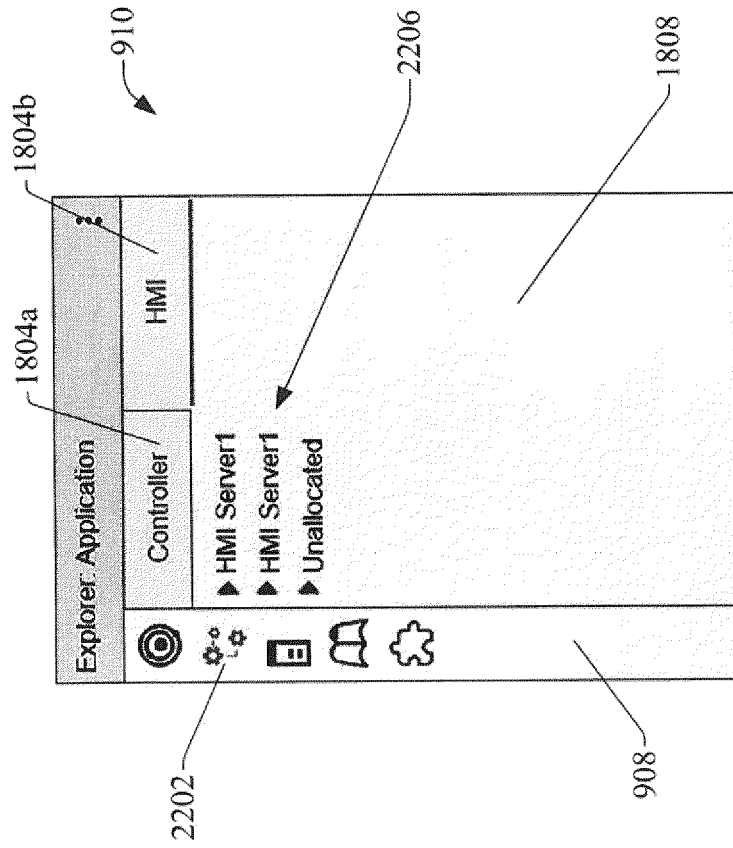


FIG. 22a

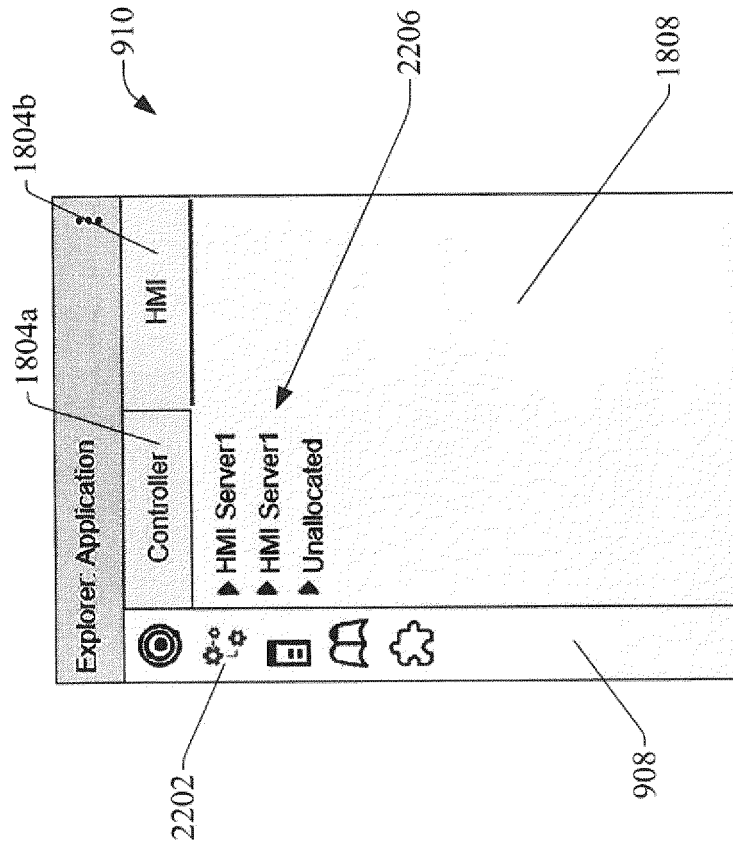


FIG. 22b

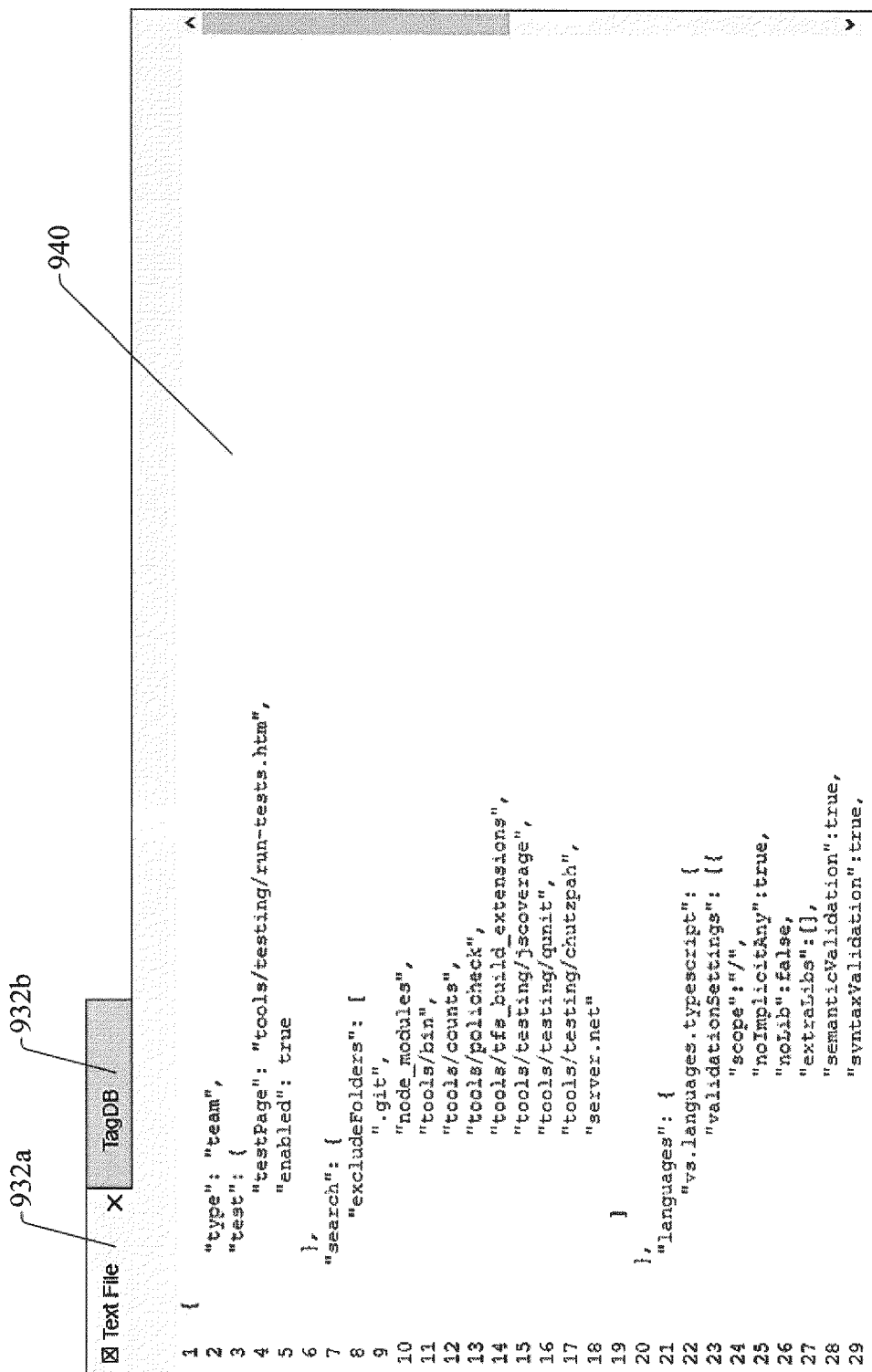


FIG. 23

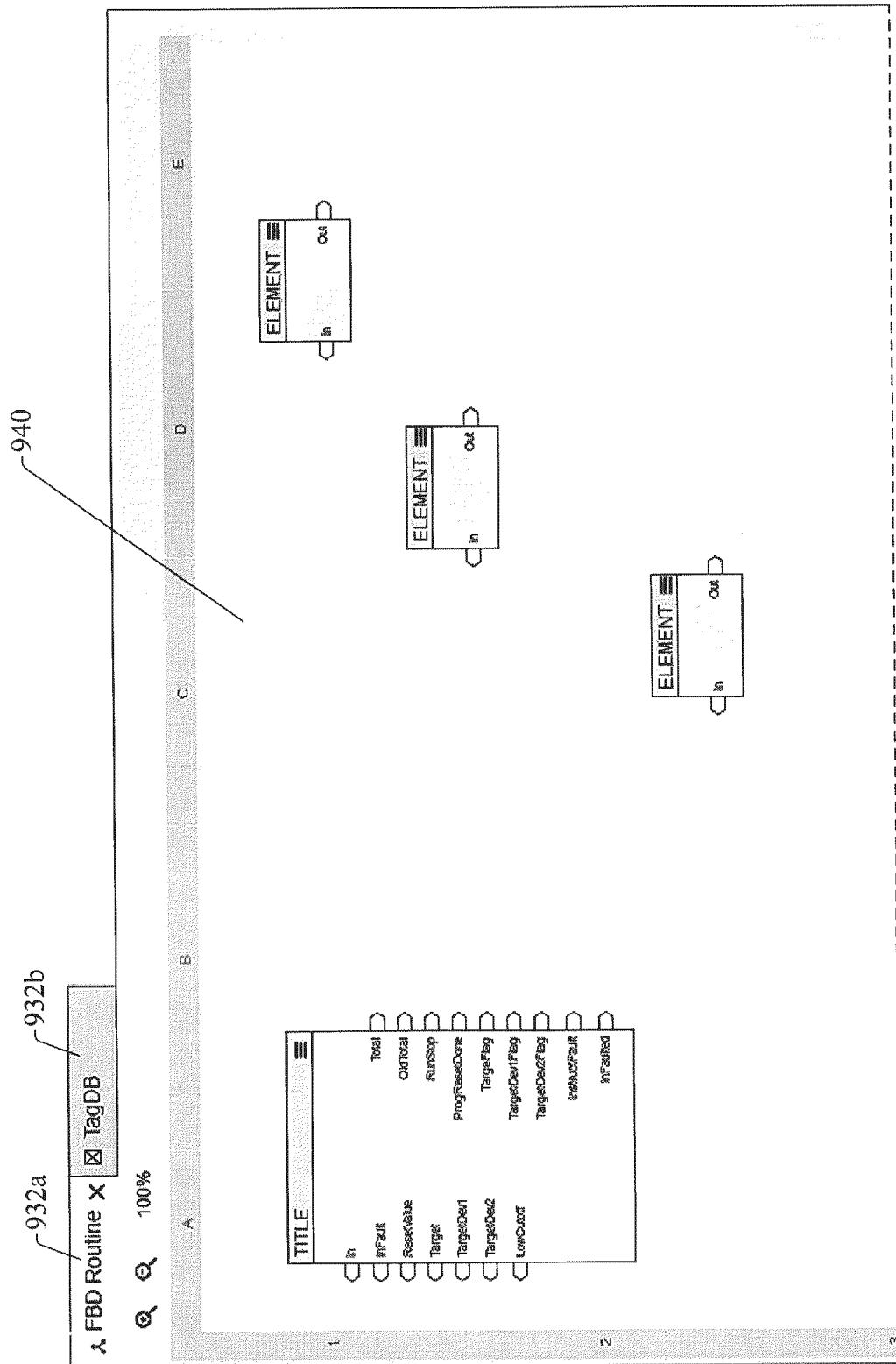


FIG. 24

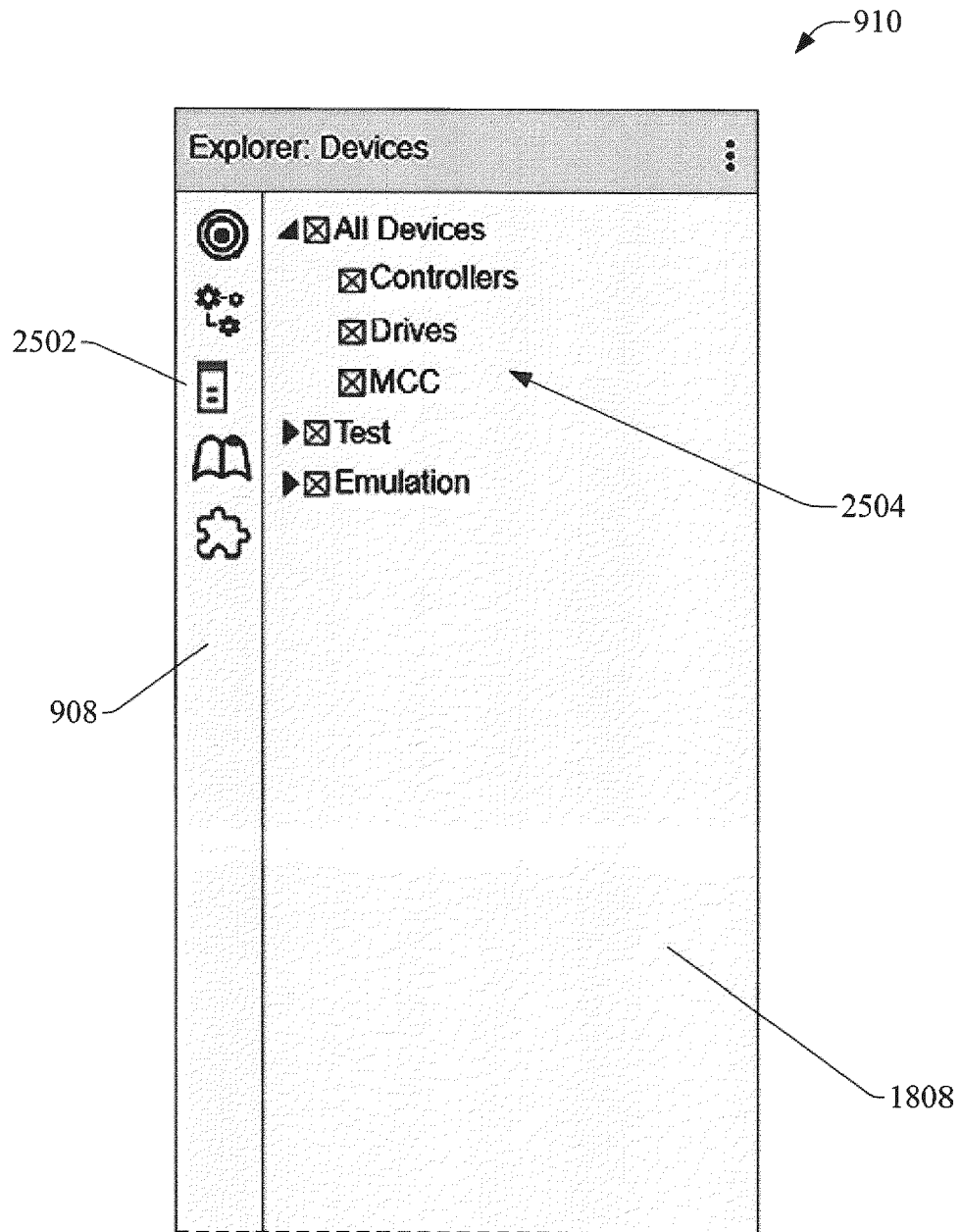


FIG. 25

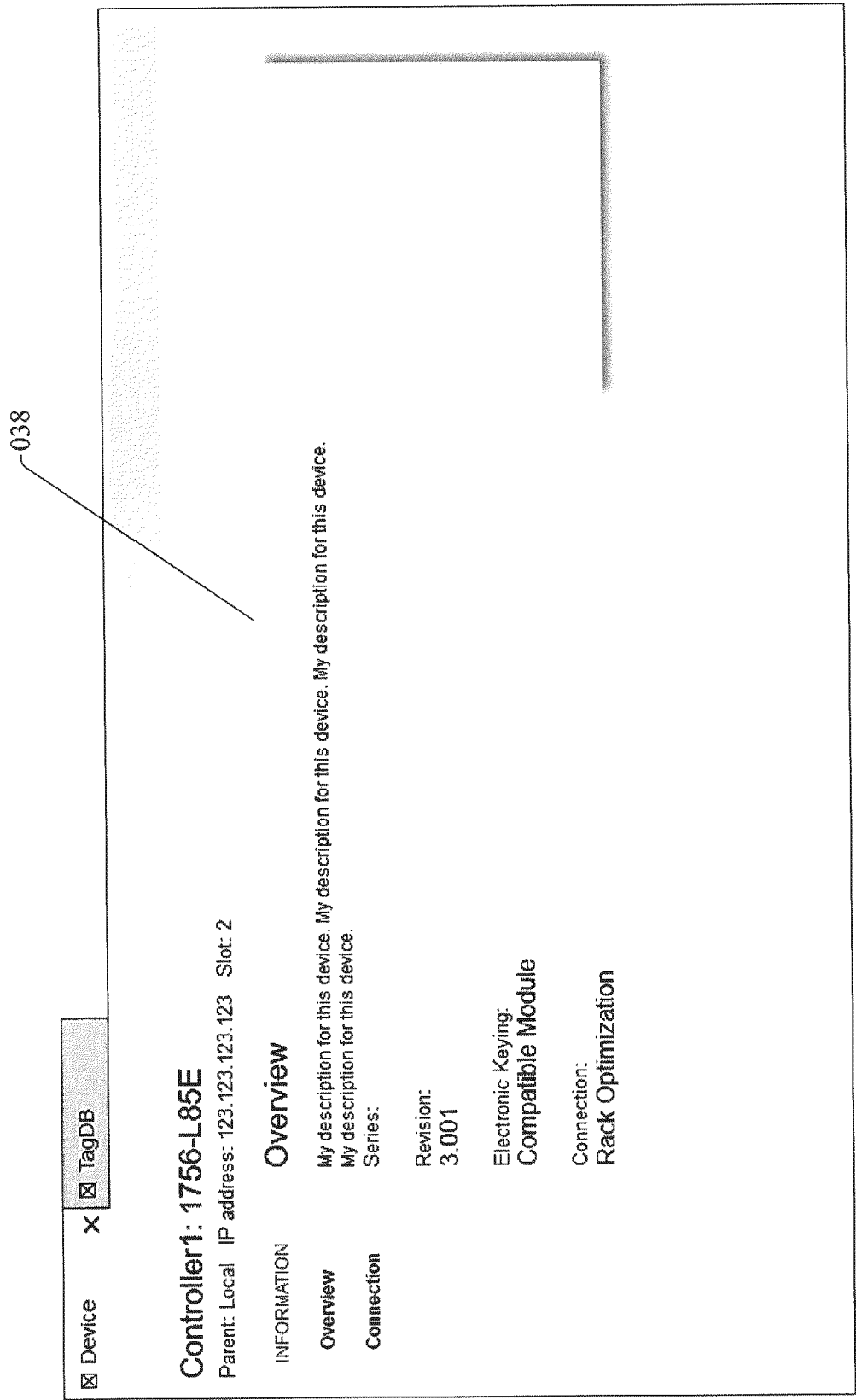


FIG. 26

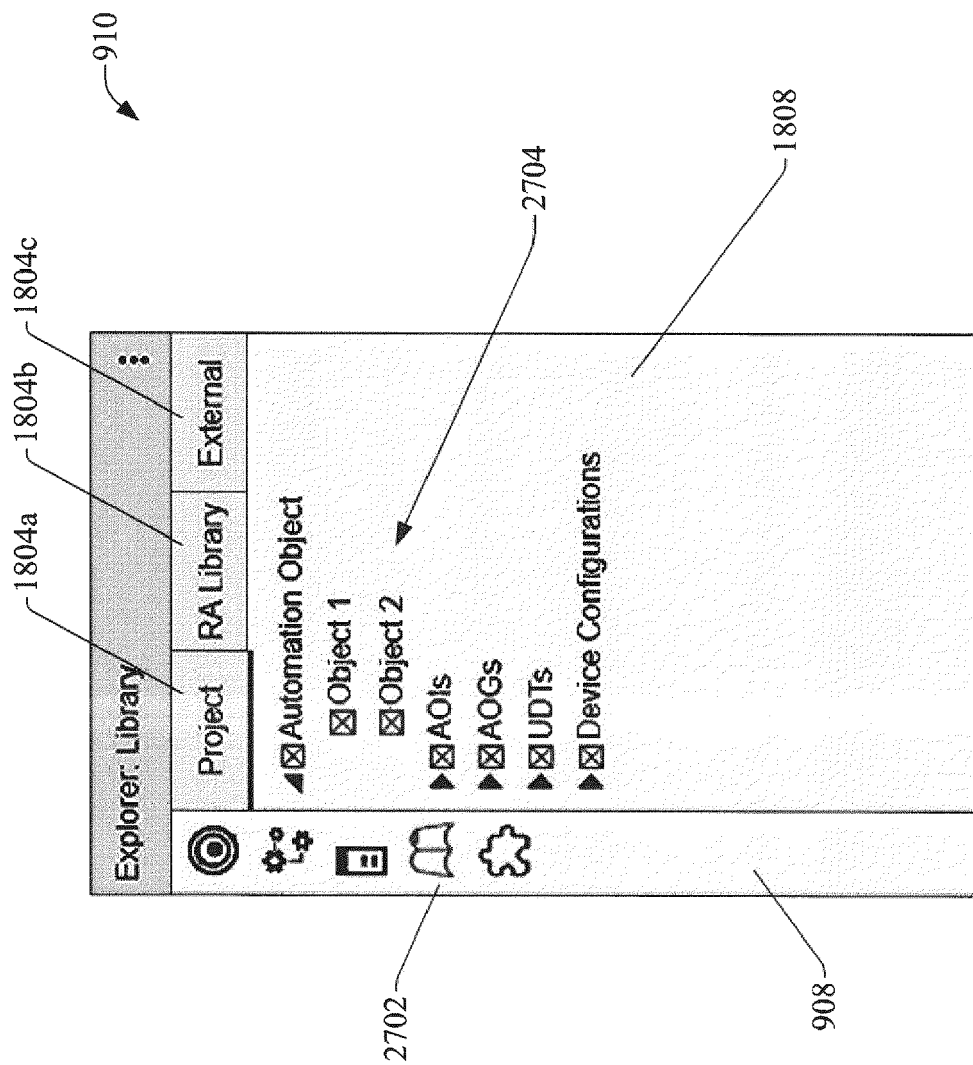


FIG. 27

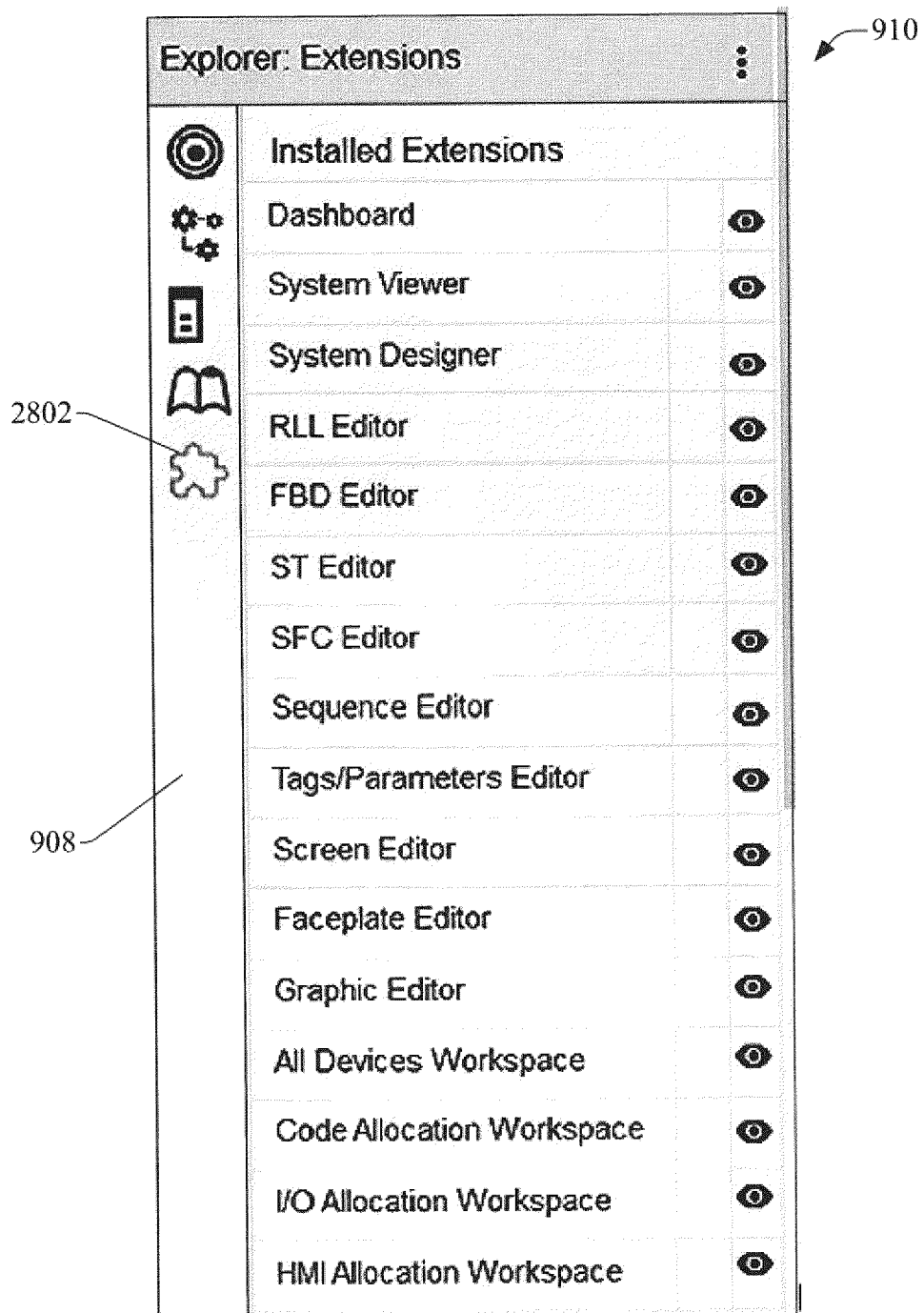
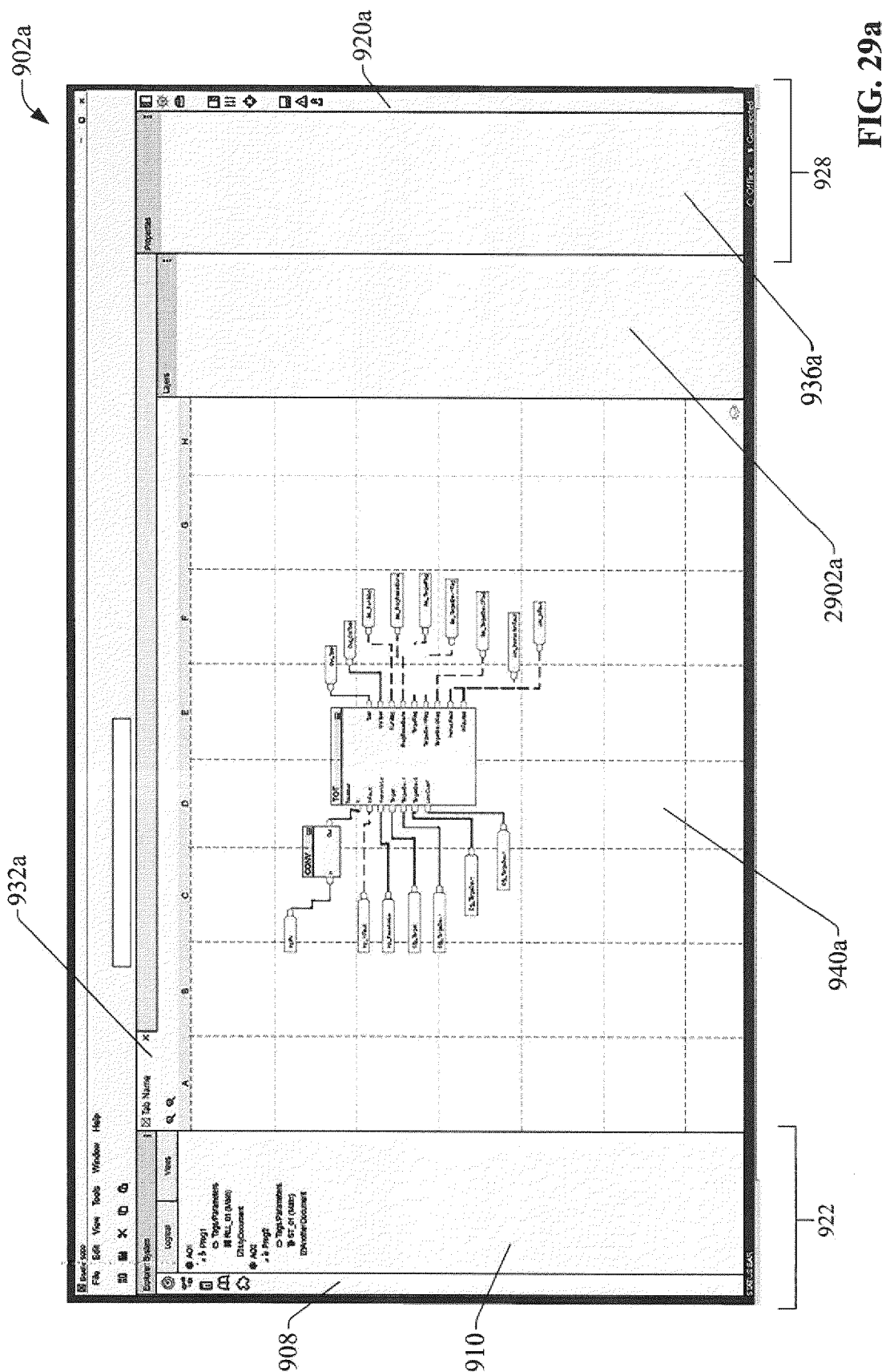
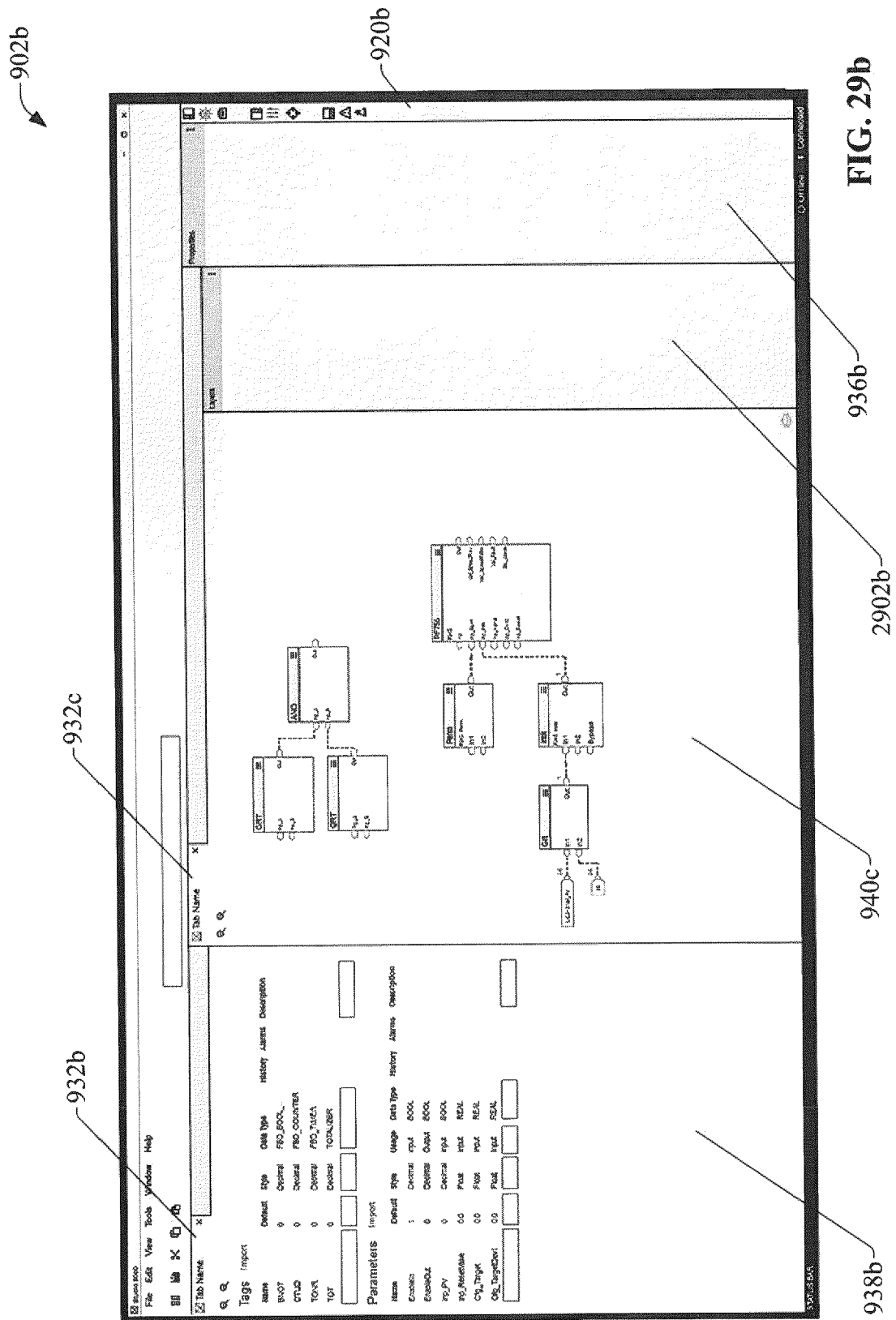


FIG. 28





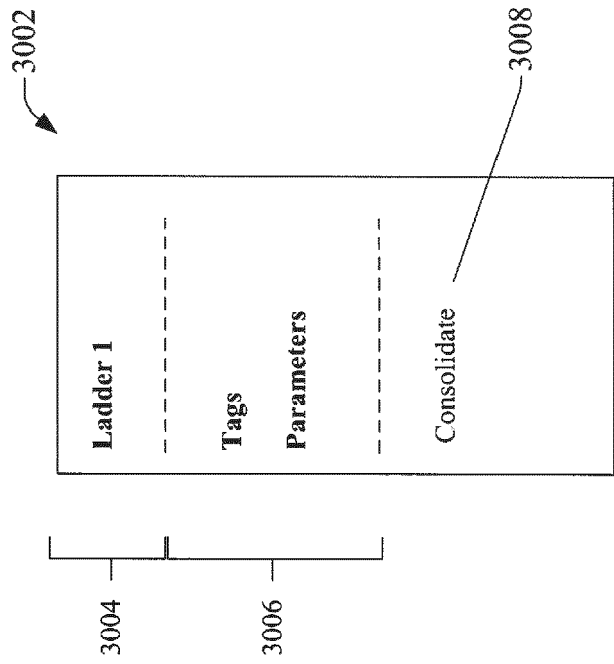
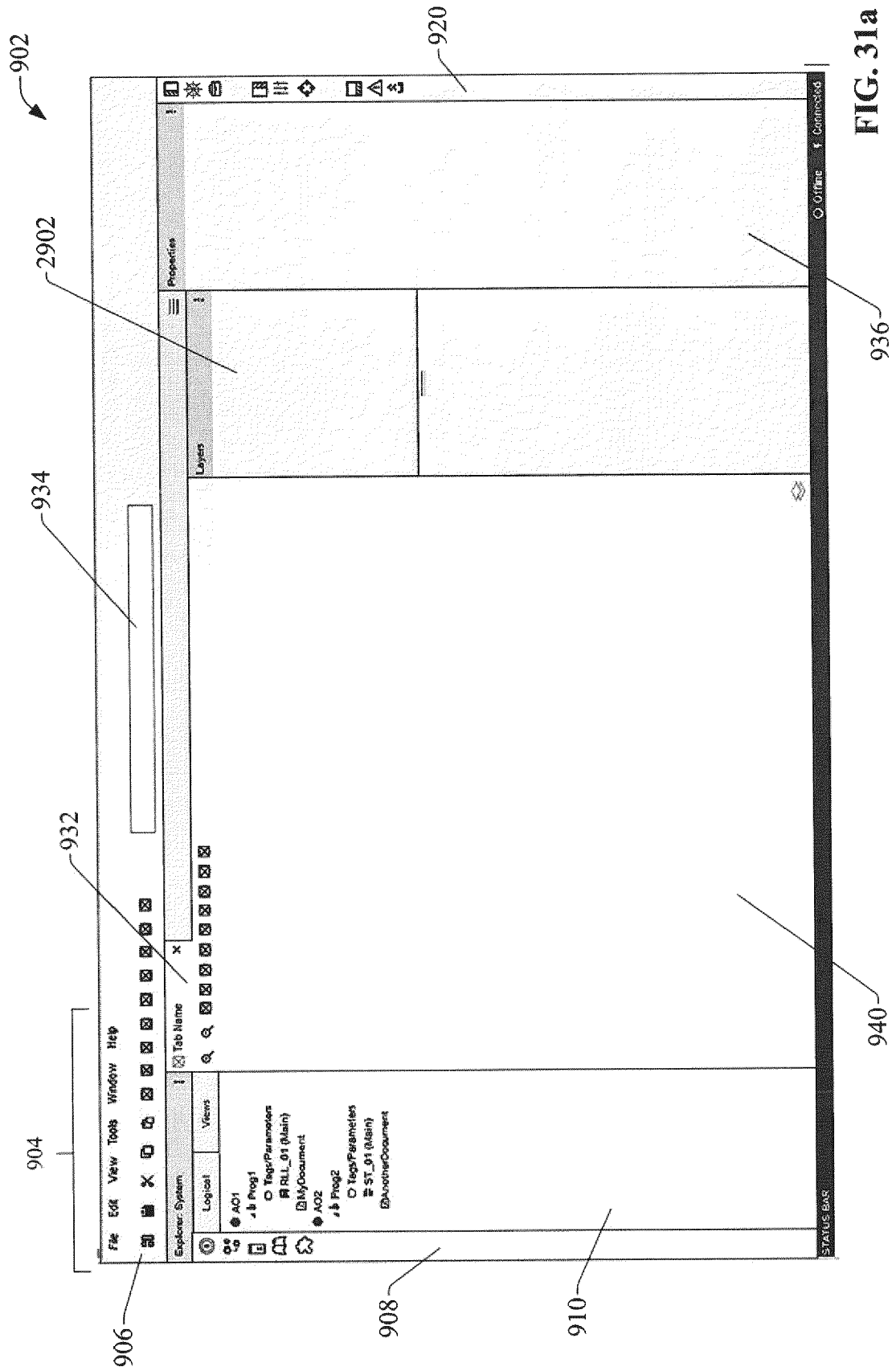
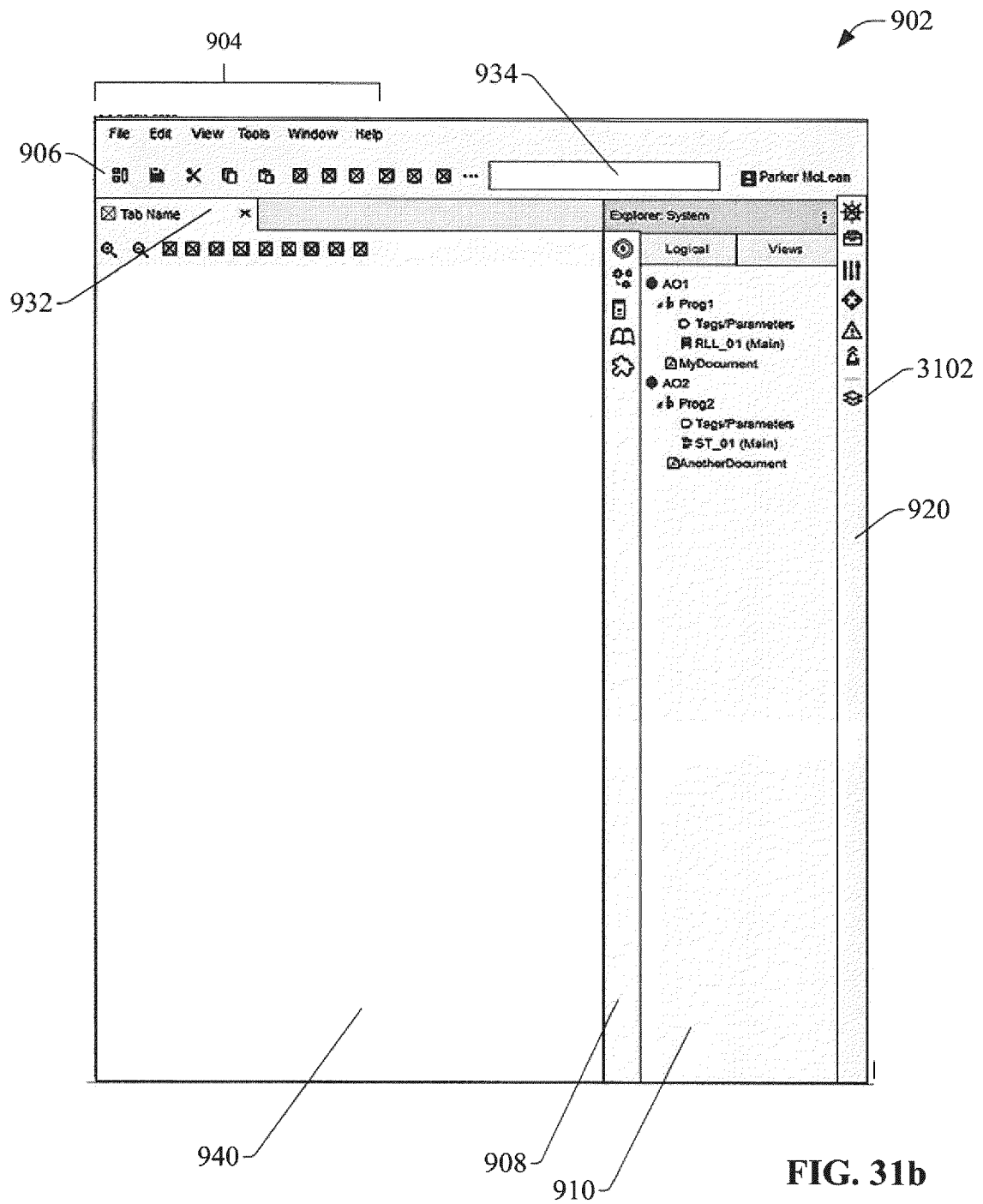
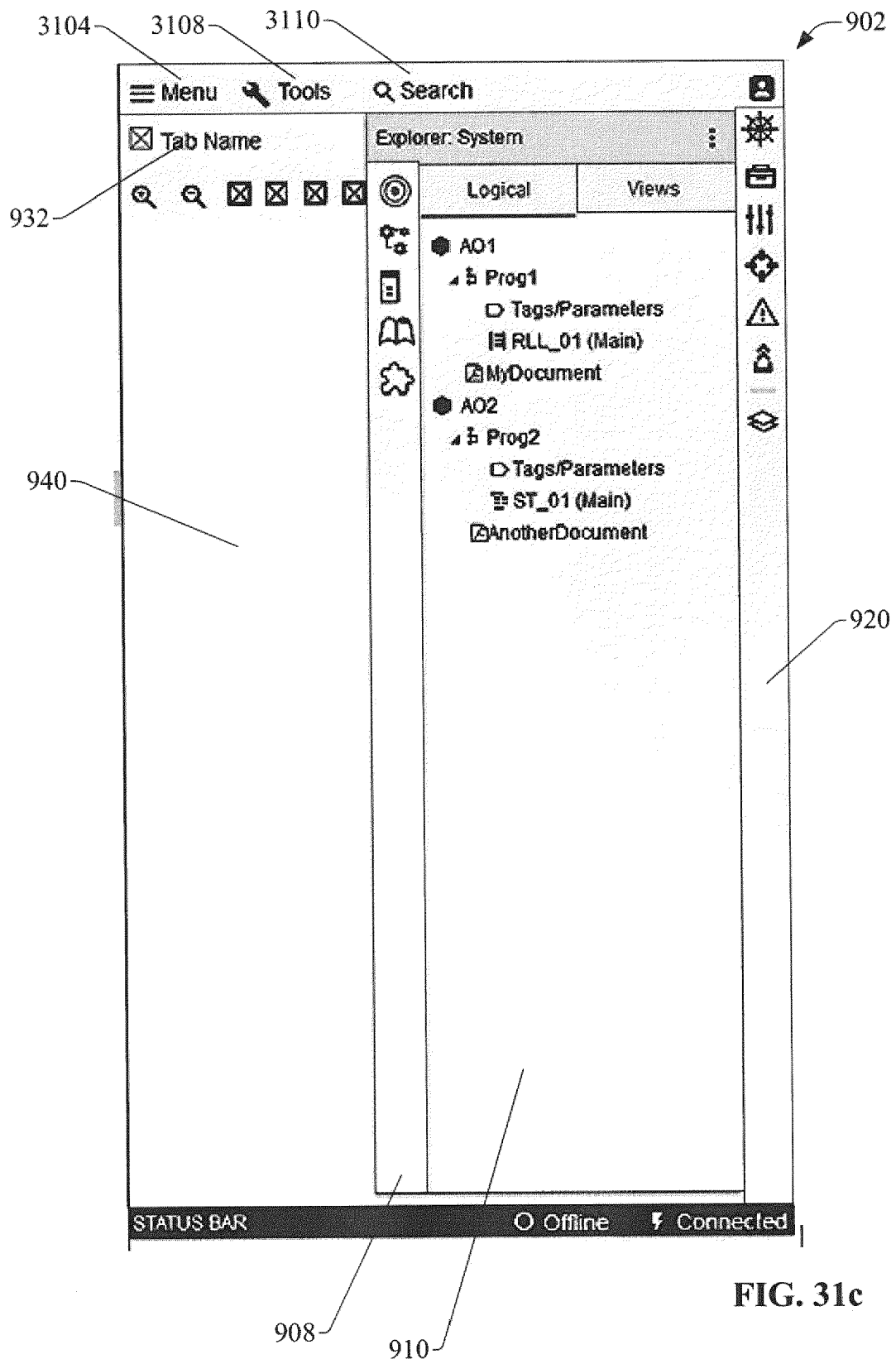


FIG. 30







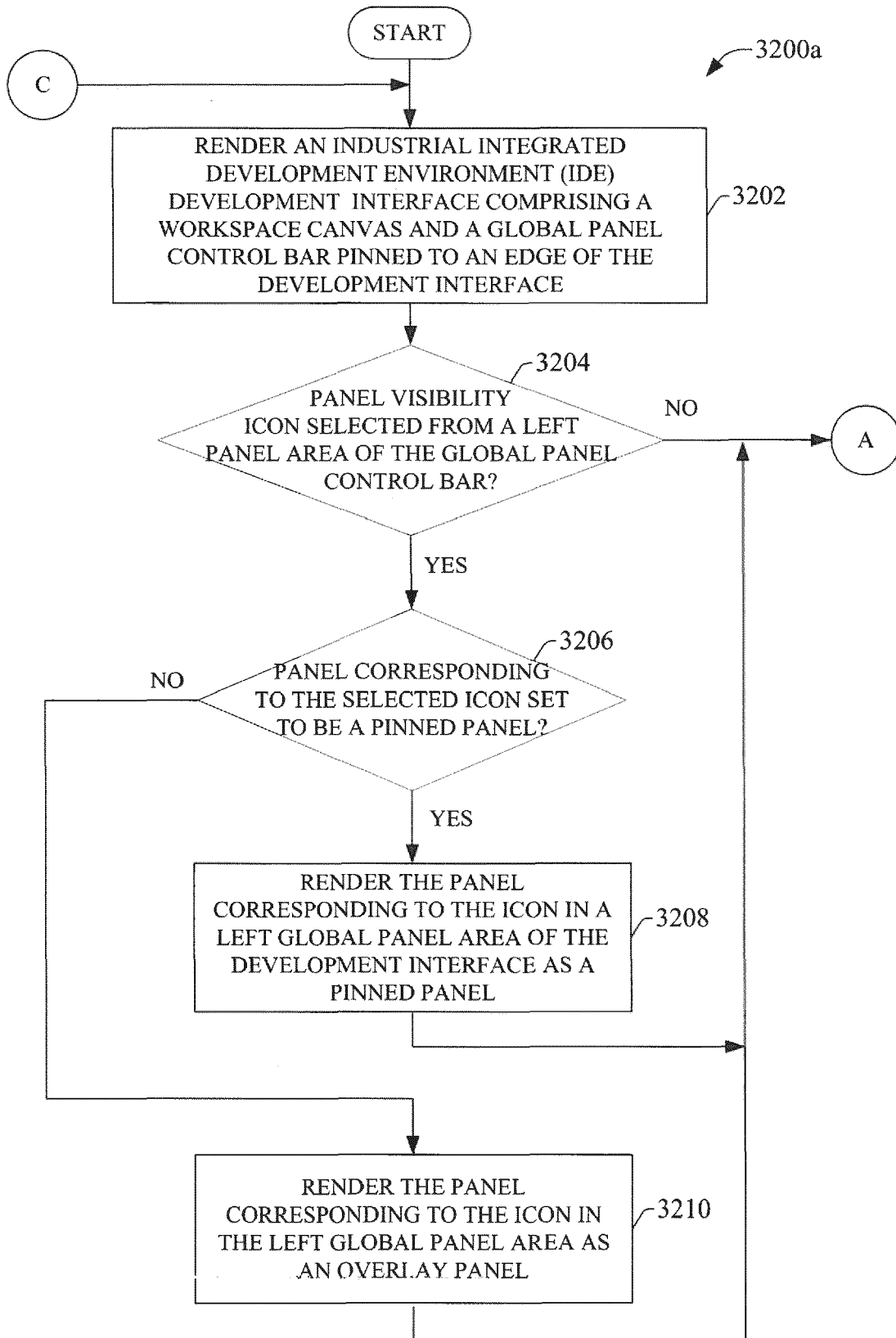


FIG. 32a

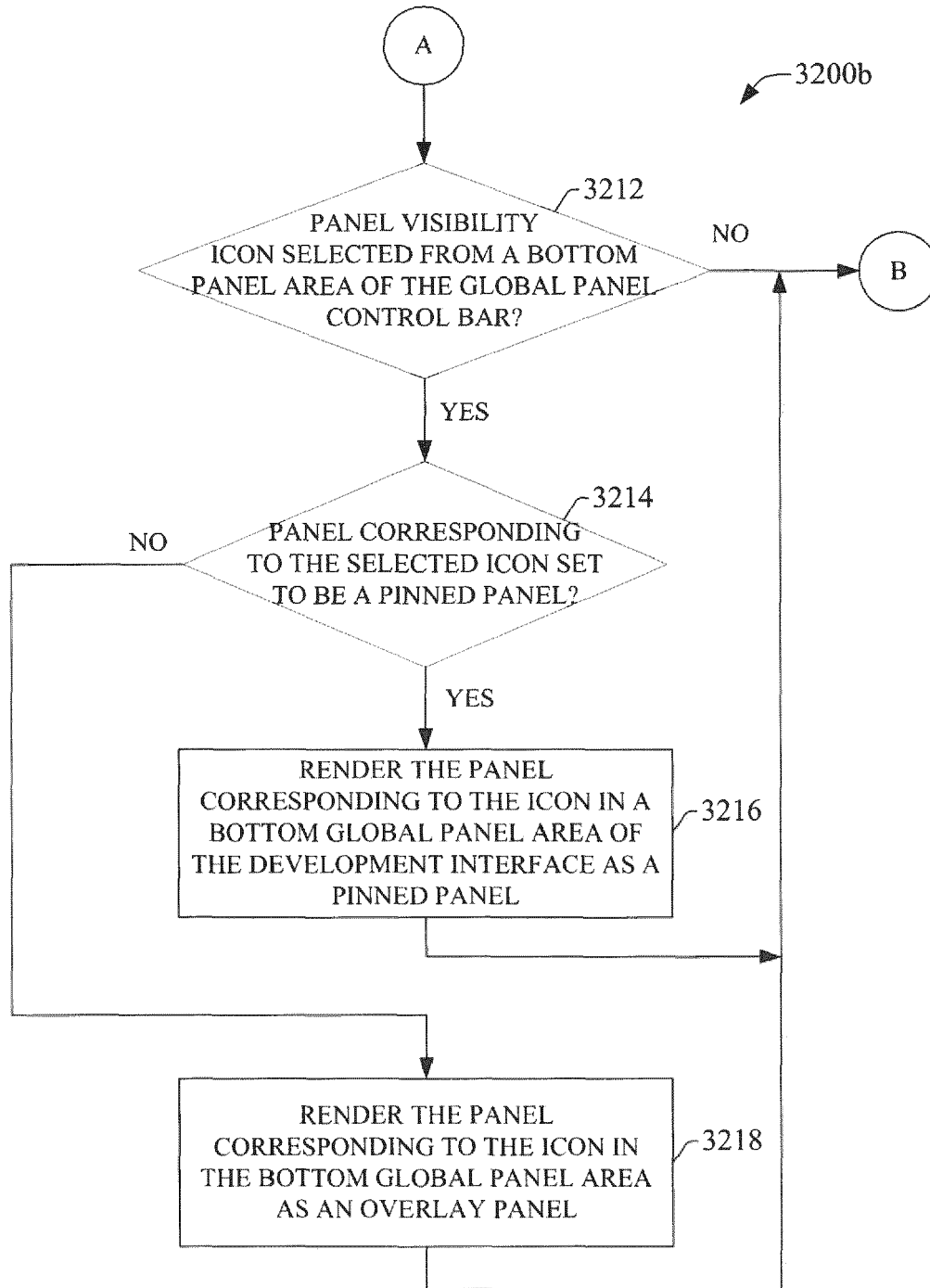


FIG. 32b

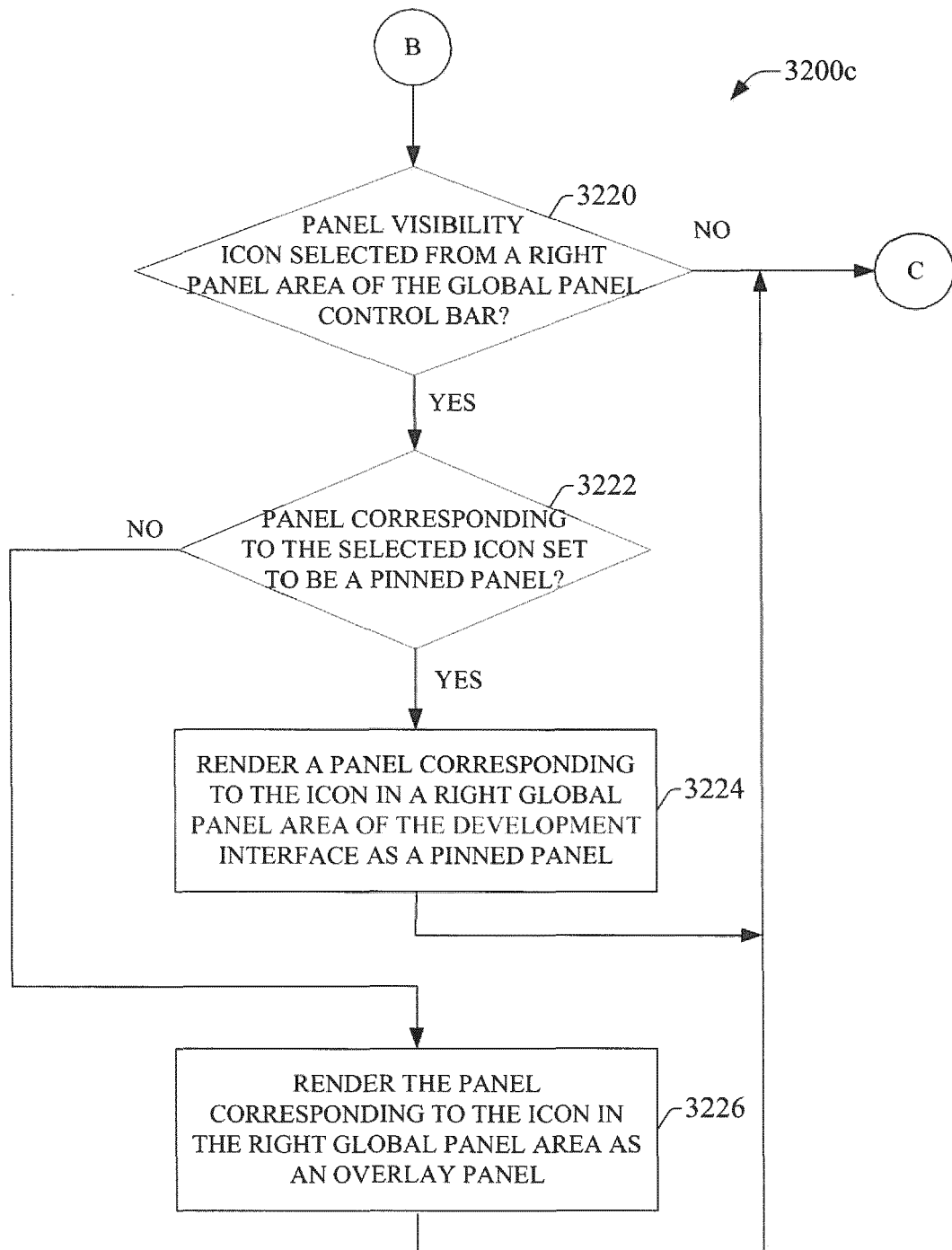


FIG. 32c

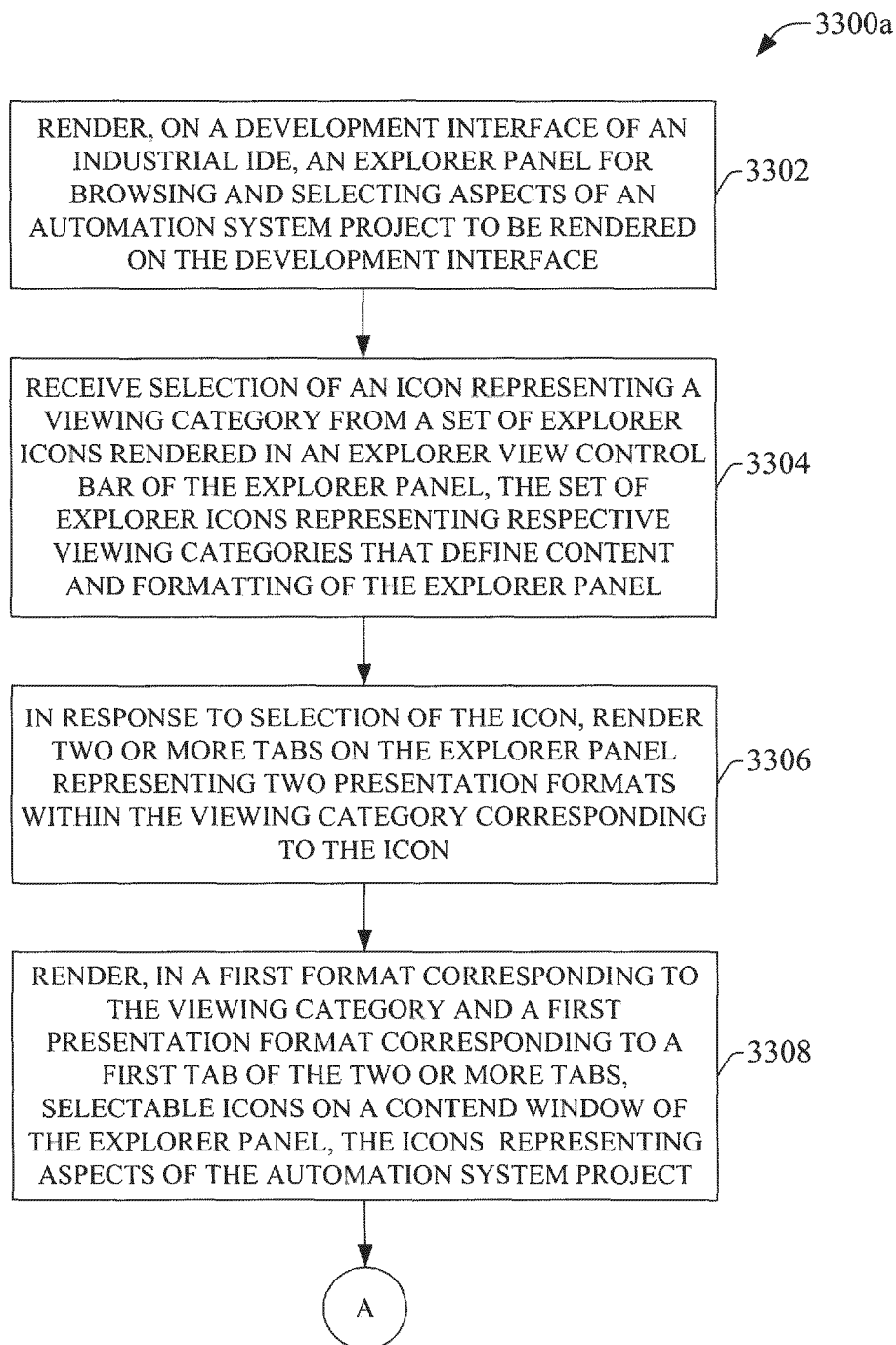


FIG. 33a

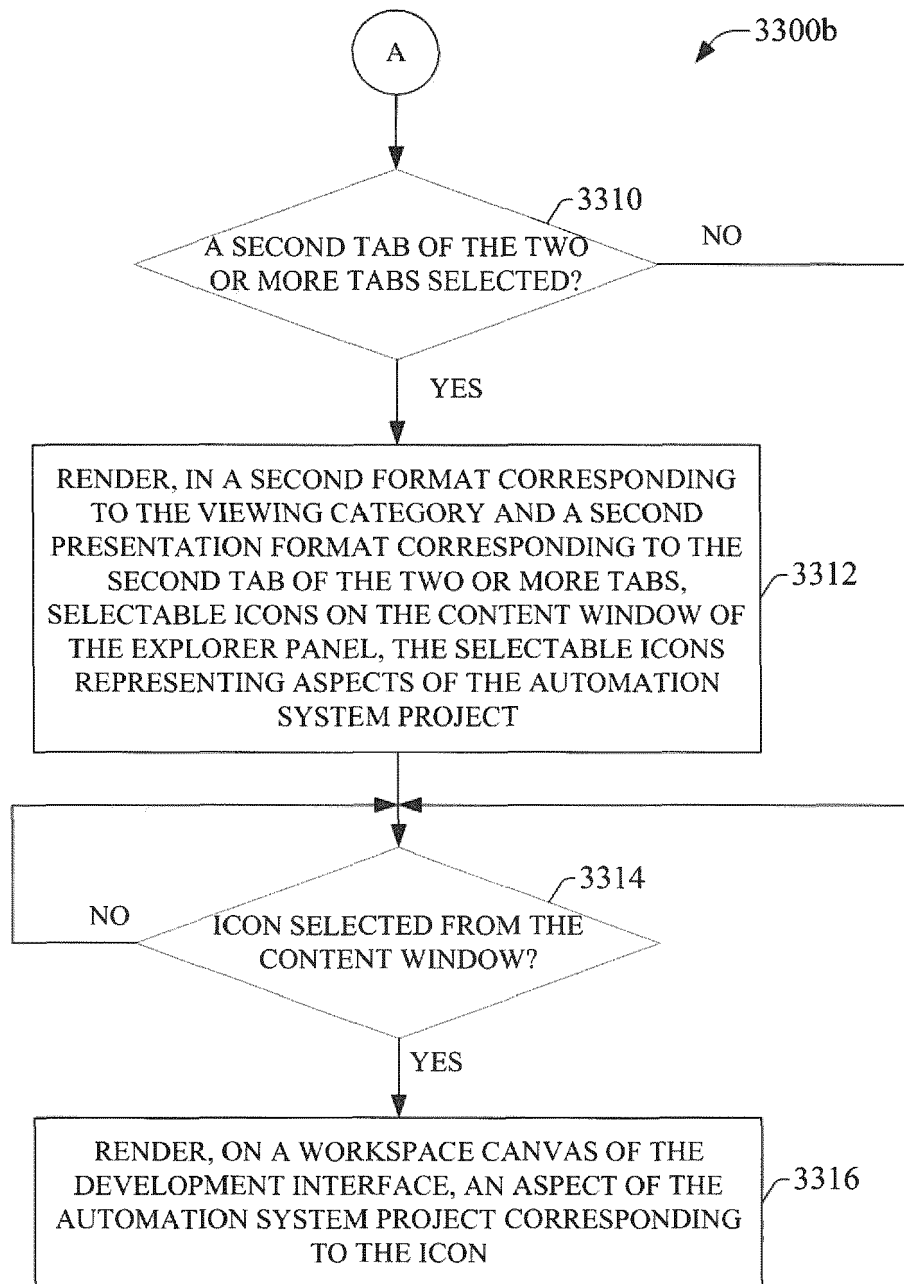
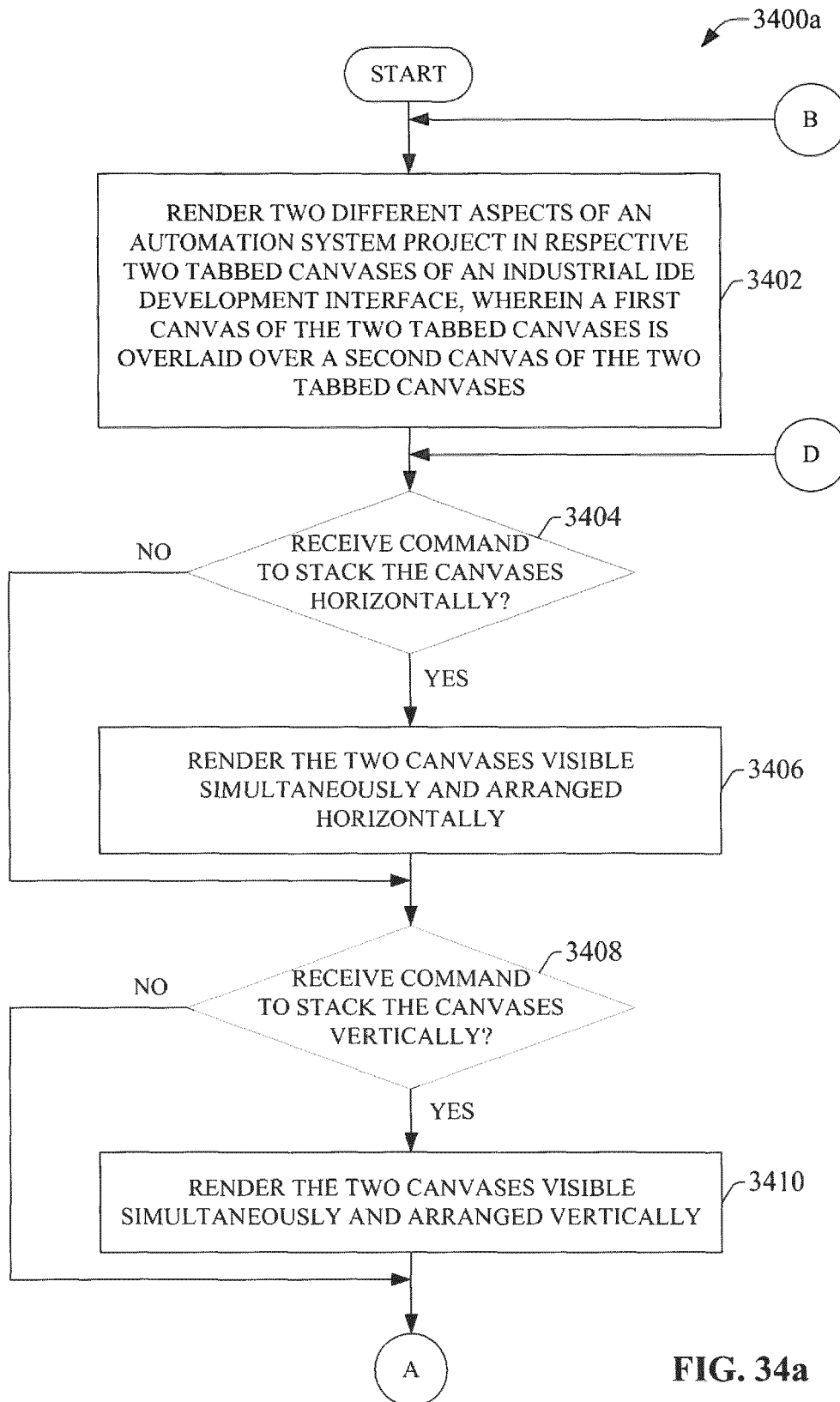


FIG. 33b



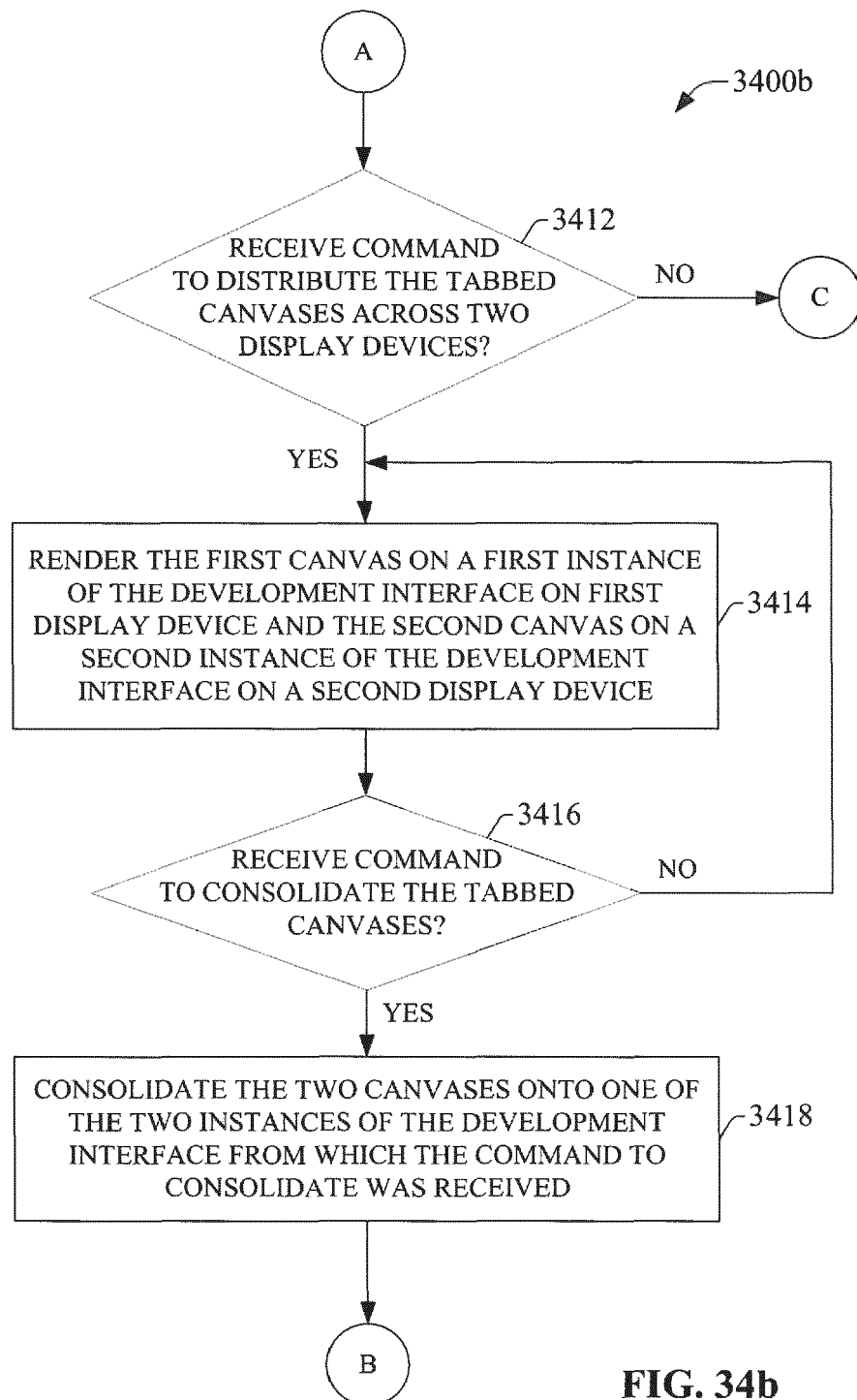


FIG. 34b

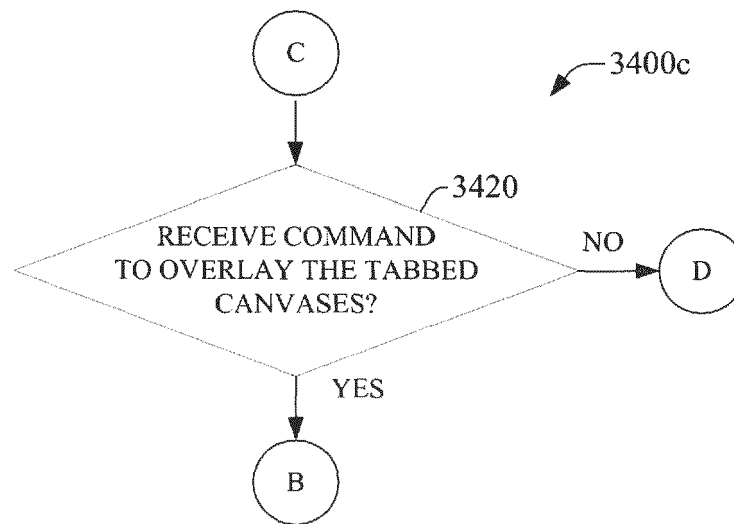
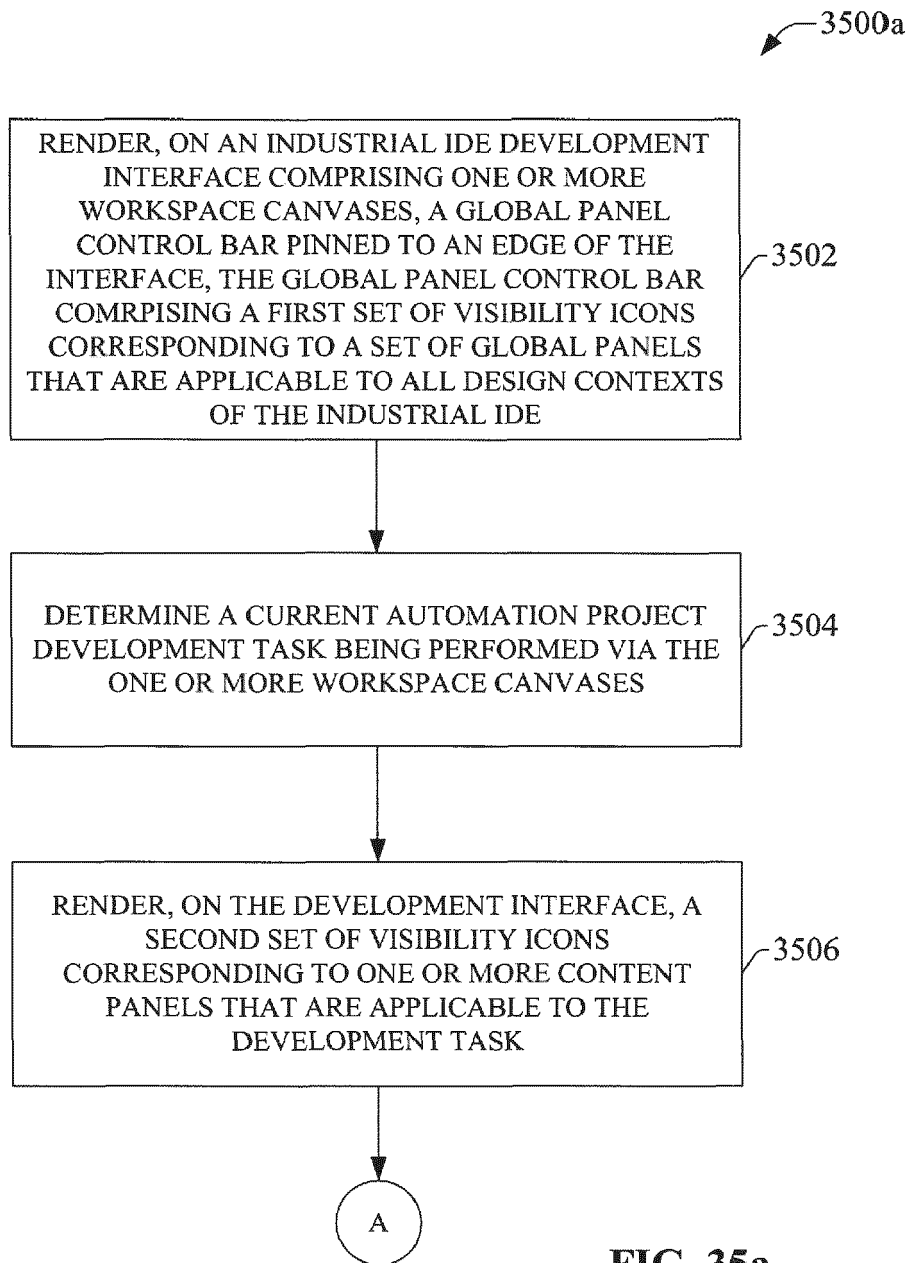


FIG. 34c

**FIG. 35a**

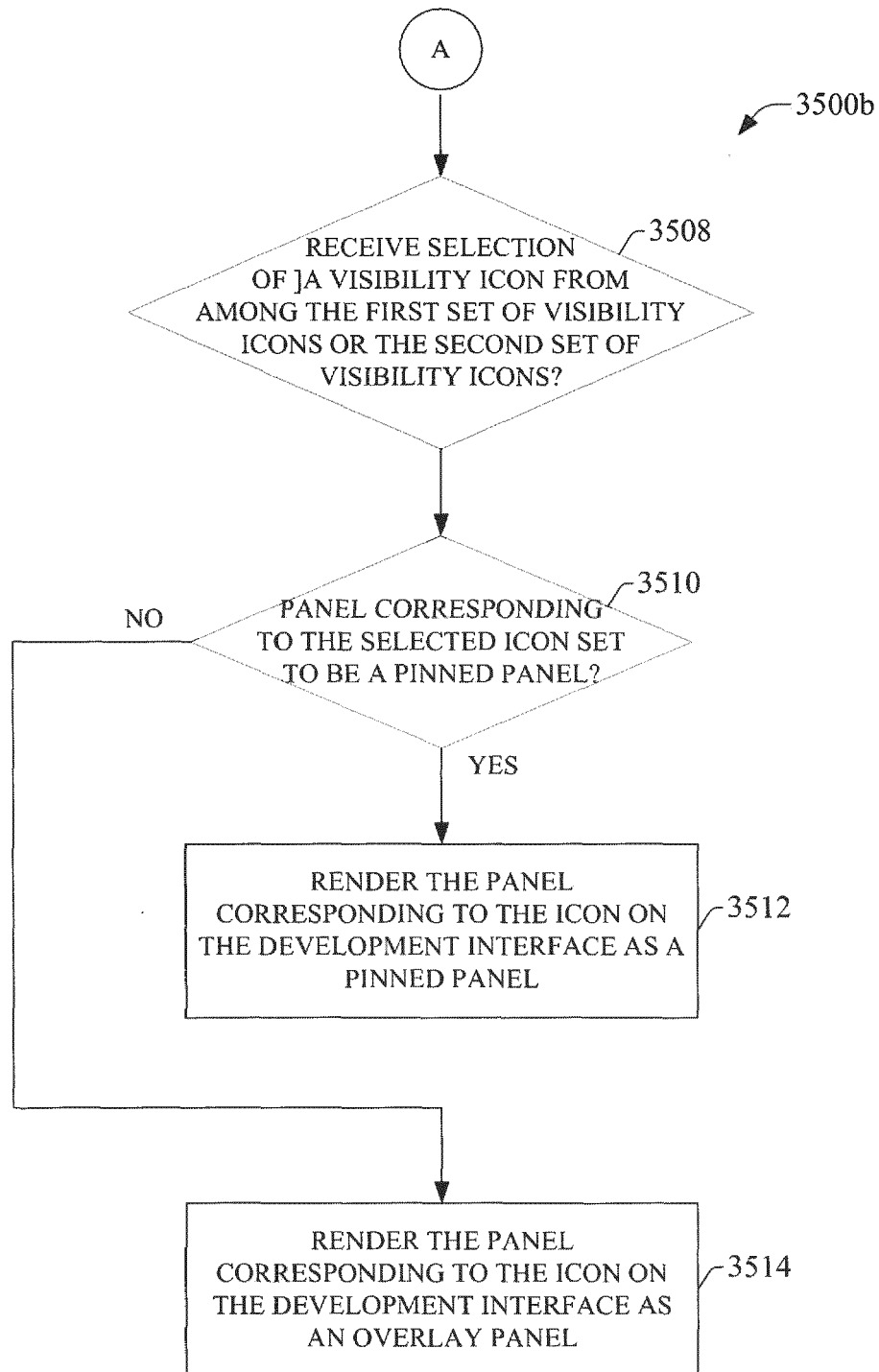


FIG. 35b

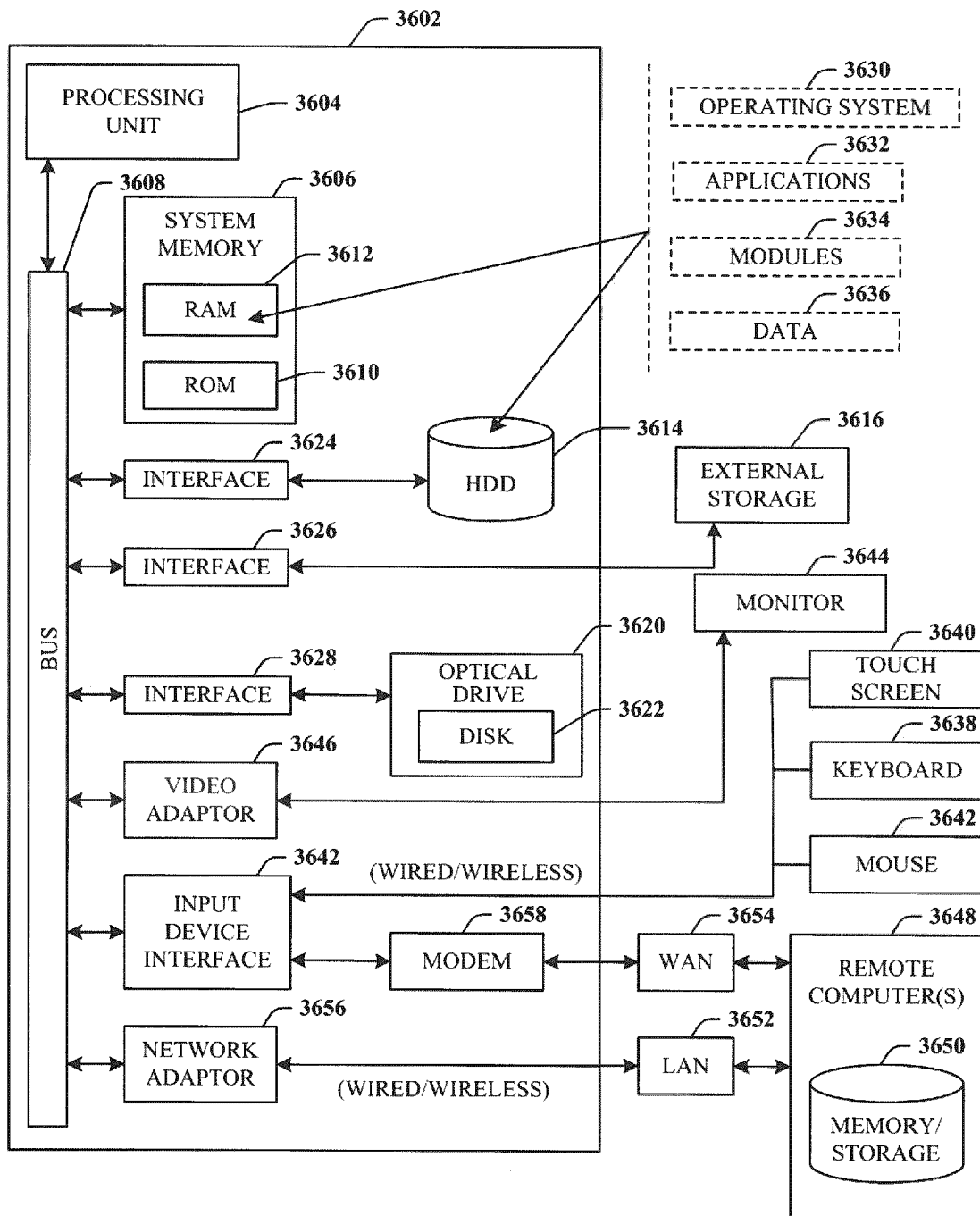


FIG. 36

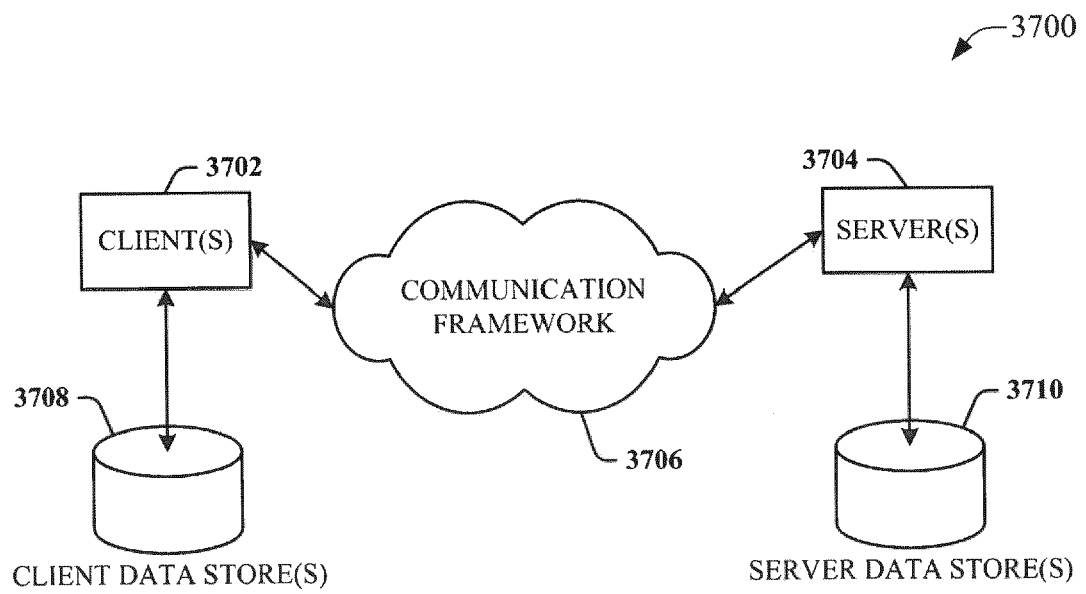


FIG. 37



EUROPEAN SEARCH REPORT

Application Number
EP 20 16 6932

5

10

15

20

25

30

35

40

45

50

55

DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	EP 3 065 017 A1 (ROCKWELL AUTOMATION TECH INC [US]) 7 September 2016 (2016-09-07) * paragraph [0003] * * paragraph [0004] - paragraph [0005] * * paragraph [0019] - paragraph [0020] * * paragraph [0021] - paragraph [0022] * * paragraph [0023] - paragraph [0024] *	1-15	INV. G05B19/042 G05B19/05 ADD. G06F8/34
X	US 5 920 717 A (NODA NATSUKO [JP]) 6 July 1999 (1999-07-06) * column 3, line 13 - column 5, line 35; claim 1; figure 1 *	1-15	
			TECHNICAL FIELDS SEARCHED (IPC)
			G05B G06F
The present search report has been drawn up for all claims			
Place of search Munich		Date of completion of the search 8 February 2021	Examiner Fulcheri, Alessandro
CATEGORY OF CITED DOCUMENTS X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons & : member of the same patent family, corresponding document			

EPO FORM 1503 03.82 (P04C01)

**ANNEX TO THE EUROPEAN SEARCH REPORT
ON EUROPEAN PATENT APPLICATION NO.**

EP 20 16 6932

5

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

08-02-2021

10

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 3065017 A1	07-09-2016	CN 105938358 A	14-09-2016
		EP 3065017 A1	07-09-2016
		US 2016259313 A1	08-09-2016

US 5920717 A	06-07-1999	NONE	

15

20

25

30

35

40

45

50

55

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82